Universidad de Alcalá
Escuela Politécnica Superior

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Técnicas de segmentación semántica aplicadas en imágenes de laparoscopia

Autor: Leticia Monasterio Expósito

Tutor: Daniel Pizarro Pérez

2018

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Técnicas de segmentación semántica aplicadas en imágenes de laparoscopia

Autora: Leticia Monasterio Expósito

Tutor: Daniel Pizarro Pérez

Tribunal:

Presidente: Luis Miguel Bergasa Pascual

Vocal 1º: Sebastián Sánchez Prieto

Vocal 2º: Daniel Pizarro Pérez

Calificación: .....

Fecha: .....

A mi familia, en especial a mis padres, a Juanma
y como no, a mis yayos...

"Nunca dejes que nadie te diga que no puedes hacer algo. Si tienes un sueño, tienes que protegerlo. Las personas que no son capaces de hacer algo por ellos mismos, te dirán que tú tampoco puedes hacerlo. ¿Quieres algo? Ve a por ello"

Will Smith, En Busca de la Felicidad

Agradecimientos

Hace poco más de año y medio fijé una nueva meta a cumplir. Hoy, pasado ese tiempo, puedo decir que la meta al fin está alcanza. Como siempre, durante este tiempo me he preguntado varias veces quién me mandó a mi meterme en un máster cuando siendo ingeniera ya podría estar tranquilamente trabajando sin necesidad de seguir estudiando. También, en meses de agobio me he preguntado lo que se me pasó por la cabeza para continuar formándome durante ochos años, pensando en la edad y la poca experiencia que podría tener gracias a seguir estudiando asignaturas poco llamativas. En verdad, la culpa no es de nadie ya que solo yo decidí dar este paso.

Hoy, puedo decir que me siento orgullosa de haber acabado el máster en las condiciones que lo empecé. Me siento orgullosa de haberme probado a mí misma y demostrarme que yo también puedo y que solo es perder el miedo que la mayoría de las veces me bloquea, bueno, en verdad he perdido parte... Pero en especial, me alegra haber dado el paso de estudiar el máster para saber definitivamente, con una confianza del 100 (%) y una varianza de $\sigma = 1/\infty$ lo que verdaderamente me gusta de un Ingeniero de Telecomunicación.

Como todo, este proceso no habría sido posible sin determinadas personas que han estado presentes en él.

En primer lugar, quiero agradecerle a Dani las oportunidades que me ha dado en este año y medio. Por haberme dejado hacer este TFM con el y por hacer que me guste más la investigación. Gracias por acercarme más aún a esa rama de teleco que sirve para ayudar. Está claro que a mi nunca me ha gustado la medicina y que desde pequeña sabía que era imposible que yo formara parte del colectivo sanitario. Pero si es verdad que siempre me he admirado a todos aquellos que de alguna forma, ponían empeño en ayudar a ese colectivo. Dani es uno de ellos y gracias a él he podido acercarme más a rama que combina ingeniería con sanidad. De verdad, muchas gracias por tus conocimientos y tus oportunidades.

Con Dani, inmediatamente tengo que dar las gracias a Marcos. Amigo desde hace ocho años y una de las personas de las que puedo decir que me llevo en esta trayectoria. A él le tengo que agradecer el hablarme de Dani. Contarme día tras día lo que hacía, sabiendo que me iba a gustar.

Dentro de la universidad, tengo que hacer una mención especial a Casillas y Fuentes. Ellos estaba claro que no podían faltar con todo el cariño que les he cogido en estos meses, para no... con la de horas que hemos echado juntos. Esta claro que sin los picos filosóficos de Casillas entre las 18:00-19:00 de la tarde. Las canciones que son capaces de crear, todas las “preguntitas” que me ha podido resolver Fuentes sobre las redes neuronales o el empeño que ha puesto Casillas en enseñarme todo lo que sabe. Sin eso, y sin ellos, estos meses no habrían sido lo mismo. No es lo mismo cerrar el laboratorio Juanma y yo los últimos a cerrarlo los cuatro. Aunque alguna noche nos hemos llevado alguna pequeña sorpresa. De verdad que hacéis que las horas se me pasen más rápido.

Fran, no podía faltar Fran tampoco. El también ha ayudado a que cuando entré en el laboratorio el proceso de adaptación fuera más rápido. Siempre está para hablar, solo hace falta girar la silla y ¡chan! Fran se gira y te da conversación.

Por último, ya fuera de la universidad sin duda tengo que darle las gracias a mis padres. Bueno, en sí no solo tengo que agradecerles el máster sino TODO. Todo lo que me han dado y lo que han hecho por mi. Por no ponerme límites nunca ni decirme que no a nada que quisiera hacer. ¿Quiero estudiar el máster? Ahí están ellos, los primeros en apoyarme y animarme a hacerlo, confiando en mí más que yo misma. En decirme que puedo en momentos de agobio máximo, por mantener la casa en silencio cada vez que lo he necesitado para no desconcentrarme y por asombrarse por todo lo que en estos años les he enseñado y contado, aunque no lo entendieran, pero ellos sabían que necesitaba enseñárselo. Eso, pocos padres lo hacen. Una vez más, este esfuerzo es de los tres. Ya no solo tenemos una carrera, también un máster y vamos a por el doctorado. ¡Vamos equipo!

Mis abuelos, a ellos les dedico el máster al igual que la carrera porque su preocupación siempre ha sido y es no verme acabar las cosas. Esta vez, al igual que el grado, lo han visto y por eso se lo dedico. Solo hay que ver la carita que ponen cuando les hablo de cosas que no entienden pero que suenan bien.

Por último, por formar parte de la universidad y de fuera de ella, por estar 16 horas, mínimo, al día conmigo. Por aguantarme días malos, buenos e intermedios. Gracias por tu apoyo, por decirme que puedo y confiar en mí. Ayudarme siempre que puedes, estar orgulloso de todo lo que consigo y hacérmelo saber por la cara que pones. Muchos dicen que compartir trabajo y vida personal con tu pareja es lo peor y que solo consigue que la relación se desgaste. Yo digo que no, porque sin ti, el máster habría sido diferente. Y esta claro que estas 16 horas diarias juntos en las buenas y en las malas nos hacen más fuerte. Porque tú eres mi otro equipo. ¡Gracias Juanma!

Y en general a todas aquellas personas que durante este año y medio, o en el total de los ocho, han estado en algún momento haciendo que la cuesta fuera más llevadera.

Resumen

Este trabajo tiene como objetivo proponer un método de segmentación semántica en imágenes médicas capturadas por una cámara de laparoscopia. La segmentación semántica consiste en clasificar cada píxel de una imagen entre un conjunto de clases, tales como instrumental, órganos del cuerpo, fondo, etc. Los métodos de segmentación semántica son una herramienta de vital importancia para la mejora, mediante técnicas de realidad aumentada, de las técnicas de cirugía medicina mínimamente invasiva (MIS) o de laparoscopia. Para ello se requieren métodos robustos a cambios en la imagen y deformaciones de los objetos y que funcionen en tiempo real. Este trabajo desarrolla y estudia métodos de segmentación semántica de imágenes basados en el uso de redes neuronales profundas, conocidas como deep learning. En concreto se estudiarán las redes de tipo "fully convolutional" compuestas por capas convolucionales y se propondrá una arquitectura basada en una encoder-decoder y capas residuales. La red propuesta, que denominaremos MIS-Net se ha evaluado en dos bases de datos de acceso público para la detección de instrumental médico y órganos en operaciones de laparoscopia de riñón en cerdos. Además se ha realizado el etiquetado y evaluación del método propuesto en una base de datos de operaciones de laparoscopia de hígado. La red propuesta se ha comparado y evaluado mediante dichas bases de datos con dos redes convolucionales del estado del arte, obteniendo resultados competitivos y permitiendo su ejecución en tiempo real en un sistema comercial basado en unidades gráficas de proceso o GPU.

Palabras clave: Realidad aumentada, segmentación semántica, zona de interés, redes neuronales, convolución, arquitectura, bases de datos, intervención quirúrgica, laparoscopia.

Abstract

The aim of this work is to propose a semantic segmentation method in medical images captured by a laparoscopic camera. Semantic segmentation consists of classifying each pixel of an image among a set of classes, such as instruments, body organs, background, etc. Semantic segmentation methods are a vital tool for the improvement, through augmented reality techniques, of minimally invasive medical (MIS) or laparoscopic surgery techniques. This task requires robust methods against changes in the image conditions and deformations of objects. It also need methods that work in real time. This work develops and studies semantic image segmentation methods based on deep neural networks, known as deep learning. Specifically, "fully convolutional" networks composed of convolutional layers will be studied and an architecture based on an encoder-decoder and residual layers will be proposed. The proposed network, which we will call MIS-Net, has been evaluated in two publicly accessible databases for the detection of medical instruments and organs in kidney laparoscopy operations in pigs. In addition, the proposed method has been labelled and evaluated in a database of liver laparoscopy operations. The proposed network has been compared and evaluated using these databases with two state-of-the-art convolutional networks, obtaining competitive results and allowing its execution in real time in a commercial system based on process graphic units or GPUs.

Keywords: Augmented reality, semantic segmentation, area of interest, neural networks, convolution, architecture, databases, surgical intervention, laparoscopy

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

ANN	Artificial Neural Network.
AR	Realidad Aumentada.
CNN	Redes Neuronales Convolucionales.
CT	Tomografía Computerizada.
FCM	Fuzzy c-means.
GD	Gradient Descent.
GT	Ground Truth.
IoU	Intersection Over Union.
IRM	Resonancia Magnética.
LoG	Laplaciano de Gaussian.
MICCAI	International Coference on Medical Image Computing and Computer Assisted Intervention.
MIS	Cirugía Mínimamente Invasiva.
MIS-Net	Minimally Invasive Surgery Network.
MUIT	Máster Universitario en Ingeniería de Telecomunicación.
SGD	Stochastic Gradient Descent.
TFM	Trabajo Fin de Máster.

Capítulo 1

Introducción

Todo logro, grande o pequeño, comienza en tu mente.

Mary Kay Ash

1.1 Presentación

Este documento presenta el Trabajo Fin de Máster (TFM) para el título correspondiente a Máster Universitario en Ingeniería de Telecomunicación (MUIT). Este trabajo tiene como objetivo el estudio y desarrollo de un algoritmo de segmentación semántica que permita, a partir de una imagen de entrada, determinar de manera automática aquellos píxeles de la misma que pertenecen a una clase concreta definida previamente: órgano, instrumental médico, fondo, etc. El contexto de este trabajo son las imágenes tomadas por cámaras de laparoscopia y, en concreto, las intervenciones de riñón e hígado.

Para ello se hará uso de redes profundas o "deep learning" como herramienta para resolver el problema propuesto de segmentación a partir de un conjunto de datos o imágenes de entrenamiento. Se plantean en este trabajo diferentes arquitecturas de redes neuronales y se evaluará su capacidad para resolver el problema de segmentación semántica con independencia del paciente y su robustez ante cambios de iluminación y puntos de vista de la cámara.

1.2 Motivación

Actualmente una de las técnicas más usadas en las operaciones quirúrgicas es la cirugía laparoscópica o también conocida como Cirugía Mínimamente Invasiva (MIS), la cual, a lo largo de los últimos 10 años, ha ido extendiéndose poco a poco hasta ponerse al nivel de la cirugía tradicional. Las técnicas MIS se basan en realizar una pequeña incisión en el cuerpo del paciente de aproximadamente entre 0.5 a 1 cm de tal forma que se pueda, con solo esa pequeña abertura, introducir un instrumento tubular conocido como puerto. Este servirá para pasar al interior del paciente, no solo los instrumentos especiales, sino además las cámaras que se usarán para captar tanto el interior como todo el proceso de operación, proporcionando de esta forma vídeos e imágenes de alta resolución [14]. En la figura 1.1 se muestra un ejemplo ilustrativo de cómo se llevaría a cabo una operación de laparoscopia.

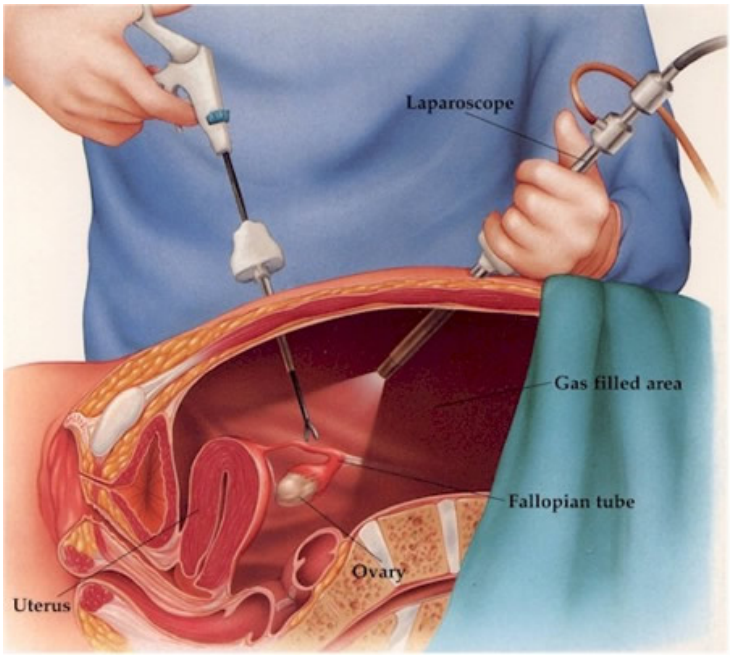


Figura 1.1: Ejemplo de Cirugía Mínimamente Invasiva

Durante la operación se monitoriza todo el proceso que está llevando a cabo el cirujano en unas pantallas situadas en el quirófano. Las técnicas MIS tienen ventajas importantes con respecto a la cirugía convencional, ya que las incisiones realizadas son de menores dimensiones, lo que hace que el paciente sufra menos dolor y que, por lo tanto, la recuperación y su correspondiente post-operatorio resulten un proceso más corto.

En lo que respecta a la preparación de la prueba, así como a todo su proceso pre-operatorio, es necesario realizar previamente una serie de pruebas que permitan al cirujano tener un control completo sobre el interior del paciente para realizar con éxito dicha intervención. Dentro de las pruebas más importantes que se suelen emplear, se encuentran algunas como la Resonancia Magnética (IRM), los ultrasonidos o la Tomografía Computerizada (CT), las cuales proporcionan al especialista la suficiente información para identificar no solo la localización de cada uno de los órganos, sino también el volumen que éstos ocupan. Lamentablemente, el empleo de pruebas de este tipo no evitan que el cirujano tenga que realizar un esfuerzo extremadamente difícil para garantizar que la operación está bajo control en todo momento.

Las técnicas MIS incorporan por tanto ventajas bastante significativas con respecto a las operaciones tradicionales, pero también implican una serie de desventajas. En estas técnicas es necesario que los médicos encargados de llevar a cabo la operación dispongan de una experiencia considerable así como de un entrenamiento lo suficientemente bueno como para suplir el hecho de no disponen de aspectos tan importantes como la percepción táctil o la visión tridimensional del interior del paciente.

Con intención de solventar el problema que la cirugía mínimamente invasiva incorpora, en los últimos años se han propuesto una serie de novedosas técnicas que permitan, mediante el uso de toda la información obtenida en el pre-operatorio y la proporcionada por las cámaras introducidas en el paciente durante la operación, fusionar ambos datos para conseguir así que el especialista disponga de una visión aumentada de su percepción durante la intervención, reforzando así el esfuerzo mental que en numerosas ocasiones tienen que llevar a cabo los médicos para poder realizar la operación. Estas técnicas que hacen uso de la información del pre-operatorio, así como de la obtenida en la fase intraoperatoria son conocidas como Realidad Aumentada (AR) [15] [16].

Sin duda, la AR en este contexto conlleva una alta dificultad, debida principalmente a la complejidad y diferente naturaleza de los datos pre-operatorios e intra-operatorios. En los primeros se describen los órganos del paciente mediante volúmenes en 3D y mediante ellos es imposible determinar la apariencia real que tendrán los órganos en las imágenes de laparoscopia. Los segundos consisten generalmente en imágenes naturales del órgano tomadas por la cámara de laparoscopia y que se ven afectadas por el movimiento de la cámara durante la fase de operación, cambios de iluminación severos, la presencia de sangrado y otros elementos que cambian drásticamente la apariencia de la imagen. Una de las tareas más importantes que requieren las técnicas de AR en este contexto es la detección automática de las regiones de interés en la imagen de laparoscopia, de tal forma que queden claramente resaltados los elementos de interés, como pueden ser los órganos y el instrumental, con respecto al resto de la imagen [17].

1.3 Objetivos

La importancia que tiene la detección automática de zonas de interés en técnicas de realidad aumentada ha hecho que el estudio de algoritmos que faciliten la segmentación en imágenes médicas a partir de cámaras de laparoscopia haya aumentado progresivamente en los últimos años. Este tipo de técnicas de clasificación de imagen a nivel de píxel se conocen como técnicas de segmentación semántica. En este trabajo se propone realizar la segmentación semántica de imágenes de laparoscopia haciendo uso de técnicas basadas en "deep learning" [18, 19]. El énfasis del trabajo es proponer una arquitectura de CNN capaz de realizar la segmentación de una imagen de entrada en tiempo real y con independencia del paciente o las condiciones en las que se tomó la imagen.

En la figura 1.2 se muestra un ejemplo de la segmentación semántica propuesta en este trabajo. A cada píxel de la imagen de entrada se asigna una etiqueta dependiendo de la clase particular a la que pertenece. En este caso las clases obtenidas corresponden a las diferentes partes del instrumental y la superficie del órgano.

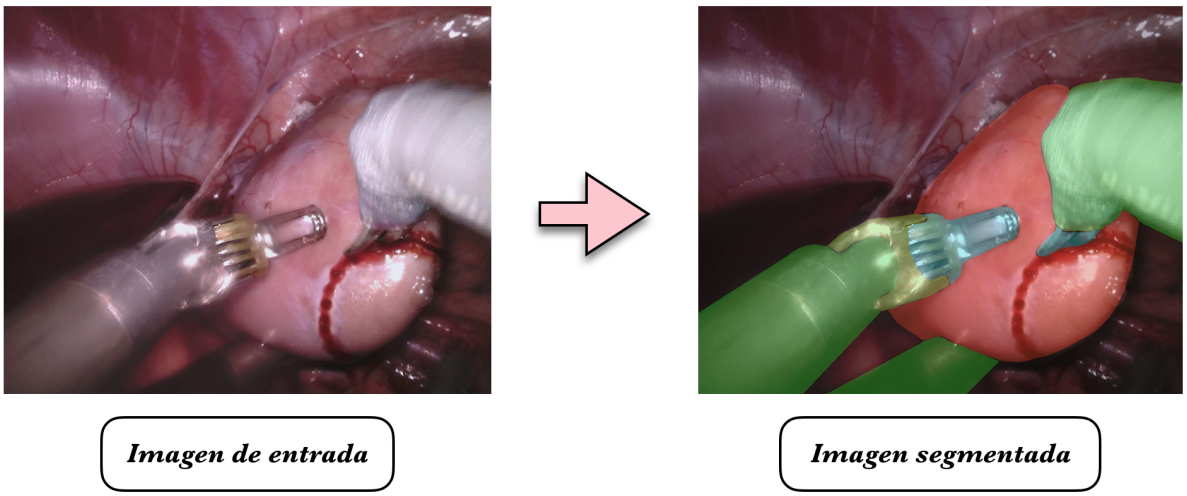


Figura 1.2: Esquema de segmentación semántica. Las diferentes clases se resaltan con diferente color en la imagen segmentada.

En este trabajo se utilizarán tres bases de datos de segmentación semántica en imágenes de laparoscopia, donde las clases a segmentar y el tipo de operación varían sensiblemente:

- **Base de datos MICCAI 2015**. Colección de imágenes procedente de intervenciones en cirugía mínimamente invasiva. Su objetivo principal es la detección de instrumental quirúrgico diferenciando para ello la parte del mango del eje, constando así de dos clases a segmentar además del fondo.

  En cuanto al origen de este *challenge*, fue creado en el 2015 para la participación en la edición de International Coference on Medical Image Computing and Computer Assisted Intervention (MICCAI) de ese mismo año.

- **Base de datos MICCAI 2018**. Conjunto de imágenes de un tamaño más extenso que el anterior donde se muestran intervenciones porcinas de cirugía mínimamente invasiva. El objetivo principal es la detección de distintas órganos de los animales así como el instrumental que interviene en la escena, constando por lo tanto de 11 clases a diferenciar: parte 1, 2 y 3 del instrumento, parénquima renal, cubierta del riñón, hilo, abrazaderas, aguja de sutura, instrumento de succión e intestino delgado.

  Este *challenge*, al igual que el anterior, fue creado en este año con objetivo de la participación en la edición de MICCAI de este mismo año.

- **Base de datos propia**. Este conjunto de datos está constituido por imágenes de operaciones en hígados a partir de laparoscopia. Está formado a partir de una serie de secuencia de diferentes pacientes en los que se ha realizado una intervención. A partir de estas secuencias, se han podido sacar diferentes *frames* que posteriormente han sido etiquetados en base a una serie de clases definidas previamente. La fase de etiquetado fue manual diferenciándose un total de 6 clases: Hígado, instrumental, ligamento, grasa, vesícula biliar y sangre.

Para visualizar un ejemplo de cada una de las bases de datos citadas anteriormente, se incorpora la figura 1.3 donde se puede contemplar el estilo de datos de los que se dispone inicialmente, así como el objetivo a conseguir en cuanto a la segmentación de dichas imágenes. Resaltar, que la escala de colores empleada en las bases de datos: MICCAI 2015 y la de carácter propio se ha visto modificado a fin de facilitar la visualización de los correspondientes *ground truth* al tratarse de colores expresados en escala de grises.

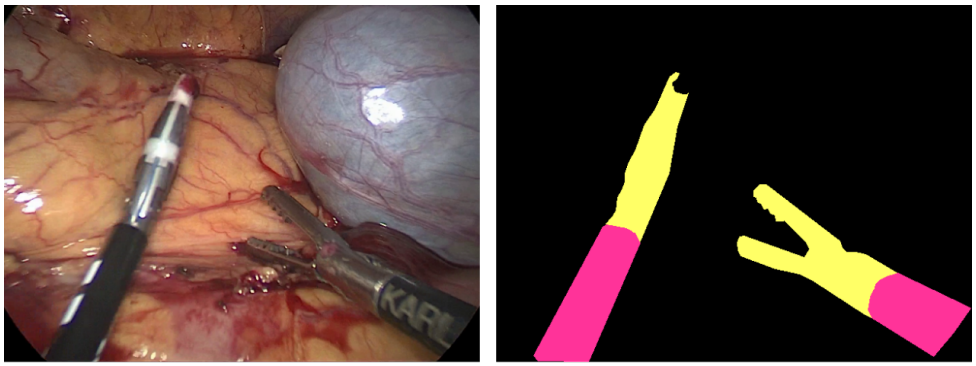
Mediante las bases de datos anteriormente descritas se propondrán métricas para la evaluación del desempeño de los métodos propuestos en este trabajo y su comparación con los métodos existentes del estado del arte.

A la hora de evaluar los algoritmos en las bases de datos utilizadas se tienen en consideración aspectos destacables como, cambios de iluminación dentro del conjunto de imágenes, posibles rotaciones y/o traslaciones de la cámara así como determinadas zonas en las que la presencia de sangre puede ser tan significativa que pueda llegar a confundir a la red a la hora de determinar las clases existentes en dicha imagen.

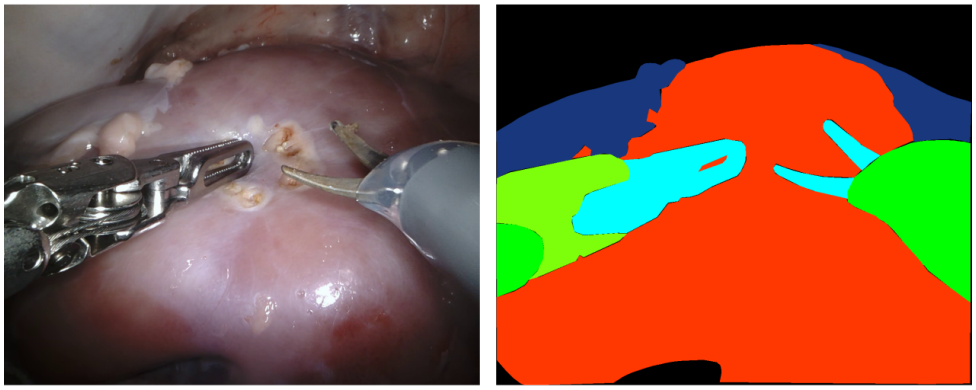
1.4 Organización de la memoria

El presente trabajo estará organizado en base a una serie de capítulos en los cuales se irán afrontando desde los conocimientos previos necesarios para llevar a cabo dicho trabajo hasta el proceso seguido para la obtención de los diferentes resultados, especificando en todo momento los detalles más significativos en cuanto a procedimiento, resultados o cualquier otro dato de interés. De esta forma, el contenido quedaría organizado en base a la siguiente estructura:

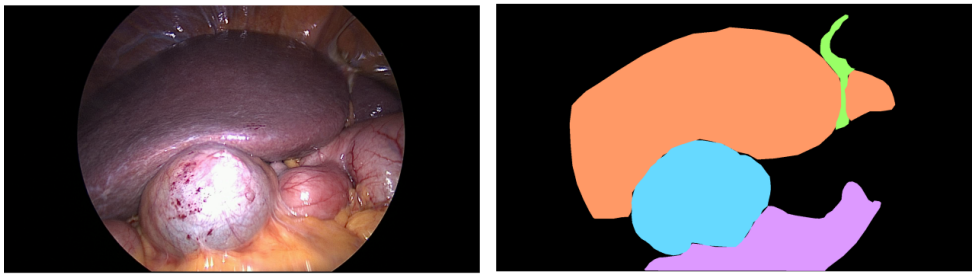
- **Capítulo 1:** Contiene una introducción al TFM desarrollado en este documento, planteando cuál es el problema que se pretende abordar así como la solución propuesta para afrontar dicho problema,



Base de datos MICCAI 2015



Base de datos MICCAI 2018



Base de datos propia

Figura 1.3: Ejemplos de bases de datos utilizadas

la cuál representaría los objetivos claros de este documento. Incluye además una descripción de los contenidos en los que se profundizará en los sucesivos capítulos.

- **Capítulo 2:** En este capítulo se detalla el estado del arte relacionado con el estudio que se va a desarrollar a lo largo de este documento. Para ello, se parte de un breve repaso por los métodos de segmentación convencionales que han sido la base durante muchos años hasta la llegada de las redes neuronales. Estas se detallan en profundidad a fin de conocer cada una de sus características así como sus fundamentos. A continuación, se detallará más en concreto las redes convolucionales, tipo específico dentro de las redes neuronales y las cuales se basan principalmente en el empleo de capas convolucionales. Finalmente, para concluir con este capítulo, se realizará un repaso por las arquitecturas más importantes, detallando aquellas que suponen, de alguna forma, una influencia a lo largo del estudio realizado.

- **Capítulo 3:** Detalla la arquitectura de red propuesta como solución al problema. Este capítulo incluye un esquema de la red utilizada donde se pueden ver las capas que conforman dicha arquitectura así como la función de coste que permite su entrenamiento. Además, se incorpora una descripción detallada de las bases de datos que se van a utilizar para la realización del estudio, explicando de forma minuciosa cada una de sus características en cuanto a tamaño, clases, procedencia y demás detales relevantes. Para finalizar, se mostrará el proceso de entrenamiento, fase compartida por cada uno de los experimentos realizados, a fin de comparar en igualdad de condiciones los resultados obtenidos por cada una de las arquitecturas probadas.

- **Capítulo 4:** Este capítulo está destinado a los resultados experimentales obtenidos una vez probadas las diversas arquitecturas de interés, incorporando para ello una serie de métricas que permita, en capítulos sucesivos, poder comparar los resultados obtenidos en cada una de las arquitecturas estudiadas. Para ello, se incorporarán tanto resultados numéricos así como visuales a fin de ver en detalle el producto de cada entrenamiento y correspondiente evaluación de las diferentes arquitecturas.

- **Capítulo 5:** Mostrará en detalle cuáles han sido las conclusiones obtenidas a lo largo de todo el estudio sobre las diferentes arquitecturas probadas. Posteriormente a este apartado, se incluye además las líneas futuras que se podrían seguir partiendo de los resultados obtenidos en este trabajo.

- **Capítulo 6:** Presupuesto donde se detallarán los gastos que se han afrontado para la realización de este trabajo.

Además de los capítulos más importantes donde no solo se plantea el estudio teórico, sino que además se detalla todo el procedimiento seguido hasta la obtención de los resultados, se incorpora además los siguientes apéndices en los cuales se puede encontrar el siguiente contenido:

- **Apéndice A:** Detalla las herramientas y recursos empleados para la ejecución de este trabajo

Capítulo 2

Estado del Arte

Existen pocas armas en el mundo que sean tan poderosas como una niña con un libro en la mano

Malala Yousafzai, Premio Nobel de la Paz 2014

2.1 Introducción

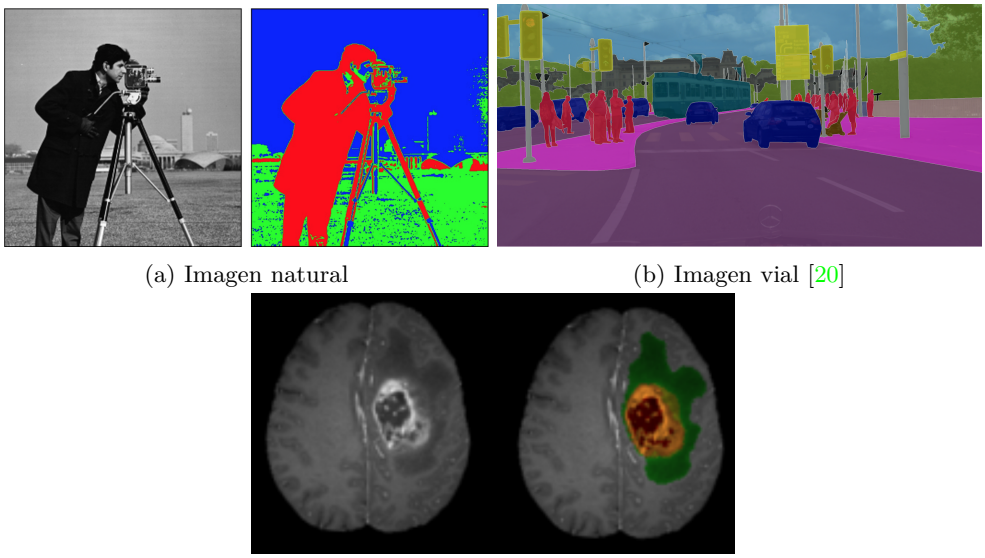
A lo largo de este capítulo se hará un breve recorrido de los métodos de segmentación semántica de imágenes más relevantes a lo largo de los últimos años, destacando de esta forma, las características más importantes de cada uno de ellos así como los fundamentos en los que se basan. Se hará a continuación incapié en la llegada de las técnicas de "deep learning" en el campo de la segmentación y como este nuevo concepto ha revolucionado la investigación,obteniendo resultados mucho mejores que los obtenidos hasta el momento por los métodos de segmentación convencionales.

2.2 Métodos convencionales de segmentación

La segmentación en imágenes es un campo de investigación importante dentro de la visión artificial, cuyos métodos han sufrido una evolución constante a lo largo de los años. Tanto es así, que a lo largo de la historia de la segmentación se pueden encontrar numerosas métodos así como diferentes tipos de aplicaciones. De carácter natural o real, ya sean paisajes o acciones humanas, escenas de tráfico o de carácter médico. En la figura 2.1 se observan algunos ejemplos de segmentación en imágenes de diferentes tipos.

En base a la literatura actual que se puede encontrar sobre los diferentes métodos de segmentación y la forma de llevar a cabo cada uno de ellos, existen diversas clasificaciones. En este caso, basándonos en la teoría desarrollada en [1] se enumerarían los siguientes tipos de métodos de segmentación:

- **Métodos basados en la intensidad**
- **Métodos basados en las regiones o similitud**
- **Métodos basados en la discontinuidad**
- **Métodos de *Clustering***



(a) Imagen natural (b) Imagen vial [20]

(c) Imagen médica [21]

Figura 2.1: Ejemplos de posible segmentación en imágenes de cualquier tipo

Así, atendiendo a la anterior clasificación, se comenzará profundizando en los métodos basados en la intensidad para terminar con todas aquellas técnicas híbridas que surgen como fruto de la combinación de varias de ellas.

2.2.1 Métodos basados en la intensidad

Los métodos basados en la intensidad son comúnmente conocidos como métodos basados en el umbral o "threshold". Esta técnica, según se detalla más en profundidad en [1] y en [22] es llevada a cabo principalmente en imágenes en escalas de grises, en las cuales se establece un umbral t con el fin de poder crear una imagen binaria en la cual se diferencien el fondo de la clase principal. De esta forma, la imagen binaria obtenida como resultado tras la segmentación se podría expresar de acorde a la ecuación 2.1 mostrada a continuación:

$$g(x,y) = \begin{cases} 1 & \text{para i(x,y)} \geqslant t \\ 0 & \text{para i(x,y)} < t \end{cases} \tag{2.1}$$

Donde $g(x,y)$ representa la imagen final obtenida tras la umbralización mientras que $i(x,y)$ sería la imagen original de entrada.En estos métodos la decisión de asignar un valor u otro para el umbral es una tarea crucial. En la literatura se describen dos formas de aplicar un umbral determinado: global o local.

En cuanto a la primera de las opciones, umbral global, consiste básicamente en establecer un único valor de t, lo que supondrá que la imagen al completo se vea afectada por ese valor. El impedimento claro de este método reside en el hecho de que normalmente los objetos en las imágenes naturales contienen una gran variedad de valores de intensidad en la imagen que dependen no solo del objeto en particular pero también de la iluminación de la escena, que en muchos casos es no uniforme. El uso de un umbral global suele dar malos resultados en general. Por su parte, la otra alternativa sería establecer una umbralización local, en la cual, la imagen original se subdividiera en regiones de menor dimensión donde se aplican valores de umbral de manera independiente.

Existen múltiples métodos para determinar el umbral de forma automática. Algunos de los más conocidos son por ejemplo umbralizar en base a una maximización de bordes o mediante el análisis de los histogramas de la propia imagen.

A continuación en la figura 2.2 se puede ver un ejemplo de posibles resultado tras diferentes umbralizaciones realizadas a una misma imagen de entrada.

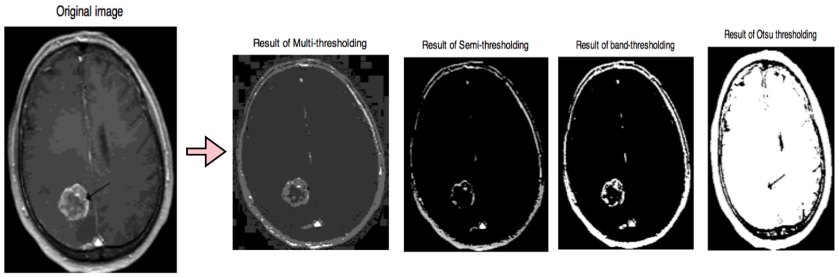


Figura 2.2: Ejemplos de posibles imágenes de salida en base al umbral establecido [1]

Para concluir, como ventajas principales de este método de segmentación de imágenes se pueden destacar el hecho de tratarse de una técnica rápida que perfectamente puede ser ejecutada en tiempo real, que además proporciona un gasto computacional bastante bajo. Como desventajas se encuentra principalmente la elección del valor de valor de t el cual puede repercutir en los diferentes resultados que se pueden obtener con la variación de dicho valor.

2.2.2 Métodos basados en las regiones

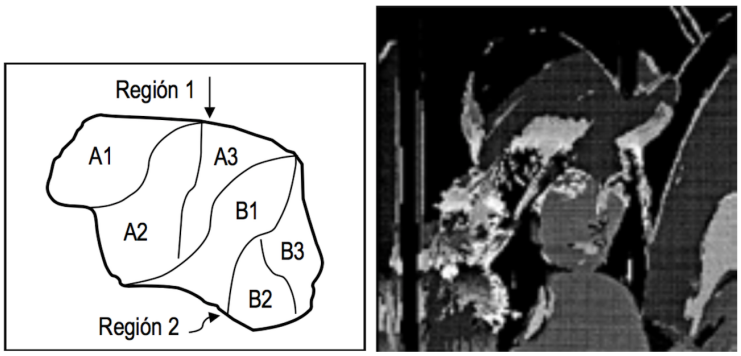
Otro de los métodos más convencionales dentro de la segmentación de imágenes es el basado en las regiones. Esta técnica se puede encontrar con mayor detalle en [1] y en [2] y consiste en dividir la imagen original en diferentes regiones cuyos píxeles comparten ciertas propiedades.

Este proceso de segmentación se puede llevar a cabo en base a dos técnicas diferentes conocidas comúnmente como *Growing region* y *Split and merge region*. La primera de ellas, *Growing region*, parte de una región semilla con un tamaño determinado que crece de manera iterativa utilizando una medida de similitud entre los píxeles de la región y los píxeles vecinos que no pertenecen a la misma. A la hora de examinar las propiedades de cada uno de los píxeles vecinos, se pueden tener en cuenta aspectos tales como la intensidad, la textura o el color. De esta forma, tras la incorporación de todos aquellos vecinos que aporten una homogeniedad a la región, se producirá el crecimiento de la semilla inicial hasta el punto en el que los siguientes píxeles vecinos empiecen a tener unas propiedades bastantes diferentes, delimitando así otra región distinta a la inicial. A continuación, en la figura 2.3 se puede observar un ejemplo de crecimiento de semilla y como progresivamente va aumentando el número de píxeles de dicha región hasta encontrar un objeto, en este caso un pétalo de la flor, a segmentar.

Figura 2.3: Ejemplos de crecimiento de regiones o "Growing region"[2]

Por su parte, el método *Split and merge region* se basa en una idea ligeramente distinta. En este caso, la imagen original es dividida de forma previa en diferentes regiones, un número determinado al inicio de la segmentación, para después analizar cada una de esas regiones de forma individual, de tal forma que posteriormente puedan sufrir divisiones o ser combinadas entre ellas. El proceso se basa en examinar los píxeles de una región determinada. Si entre ellos hay ligeras diferencias en cuanto a propiedades, esa región será dividida en subregiones más pequeñas, de tal forma que cada una de ellas cumpla un criterio de homogeneidad. El caso contrario también podría darse, fundiendo varias regiones en una región suficientemente homogénea. El objetivo claro de estos dos métodos es, al igual que el crecimiento de regiones, el de garantizar una segmentación en la imagen donde los diferentes objetos de la imagen se encuentren perfectamente separados.

A continuación, en la figura 2.4 se observan dos imágenes que representan el proceso de segmentación por regiones de tipo *Growing Region*. En la figura 2.4a se puede observar un ejemplo del método *Split and merge region* donde se subdividen dos regiones iniciales, incrementando el número total de regiones. Por su parte, la figura 2.4b representa el resultado obtenido tras la segmentación de una imagen determinada mediante el método de las regiones.



(a) Ejemplo de subdivisión de regiones

(b) Imagen segmentada según método de regiones

Figura 2.4: *Split and Merge regions* [2]

Para concluir, los métodos basados en regiones son capaces de generar segmentaciones de gran calidad en determinados tipos de imagen en comparación con otros métodos convencionales. Sin embargo, tienen como inconveniente fundamental la dificultad existente en la selección correcta de la semilla y los criterios de homogeneidad de las regiones.

2.2.3 Métodos basados en la discontinuidad

Otro de los métodos de segmentación bastante relevante en este campo es el basado en las discontinuidades, el cual se encarga de llevar a cabo una división de la imagen en base a los cambios bruscos en niveles de grises que se puedan dar en dicha imagen. Estas discontinuidades normalmente pueden ser detectadas en base a la localización de puntos aislados, de líneas o de detección de bordes y contornos de la propia imagen, siendo esta última la técnica más común a la hora de realizar una segmentación basada en la discontinuidad. Es por eso que a lo largo de esta sección se centrará todo el interés en la detección de bordes o *Edge detection*, por ser los encargados de delimitar de forma más clara dos regiones vecinas y claramente diferentes entre sí en cuanto a niveles de grises. De esta forma, a partir de los conceptos desarrollados en [1], [2] y [23] se puede hacer una clasificación de bordes en base a su anchura, el ángulo de su pendiente de variación y las coordenadas de su punto medio, pudiendo de esta forma clasificar los

bordes como tipo línea, escalón, rampa y tejado. Esta clasificación se puede observar a continuación en la siguiente figura 2.5.

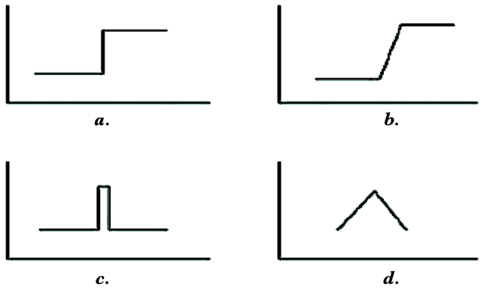


Figura 2.5: Clasificación de bordes [1]

Una vez mencionada la forma de clasificar los bordes, es importante saber cómo identificar esas discontinuidades. Para ello se hace uso de las derivadas de primer orden usando para ello el operador gradiente y de segundo orden donde se encontraría el operador laplaciano, en los cuales se profundizará a continuación.

2.2.3.1 Derivada de primer orden: Operador Gradiente

El operador gradiente es sin duda, una de las técnicas más utilizadas para la detección de bordes en 2D. Su funcionamiento se basa principalmente en la realización de las derivadas parciales de primer orden de la función $f(x, y)$, siendo f la imagen a segmentar. Por lo tanto, el gradiente sería el resultado de $\delta f/\delta x$ y $\delta f/\delta y$.

Existen varios operadores que hacen uso de la derivada de primer orden para llevar a cabo la detección de bordes. Entre ellos, se pueden encontrar el operador *Robert, Prewitt* y el operador *Sobel*. Todos ellos se caracterizan básicamente por el empleo de unas máscaras tanto para la componente x como para la componente y. La diferencia reside en el tamaño de éstas y la forma que tienen. De este modo, el operador *Robert* al ser el más sencillo de todos, emplea para la detección máscaras de tamaño 2 x 2 mientras que el resto lo hacen con máscaras de dimensiones 3 x 3. En cuanto a la forma de cada una de ellas se pueden observar a continuación en las ecuaciones 2.2, 2.3 y 2.4 donde corresponden con los operadores *Prewitt, Sobel* y *Robert* respectivamente.

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.4}$$

2.2.3.2 Derivada de segundo orden: Operador Laplaciano

El empleo de operadores de segundo orden se basa en la búsqueda del cruce por cero de las segundas derivadas parciales del gradiente, con el objetivo de localizar los máximos locales de la función $f(x, y)$

para tratarlos posteriormente como bordes de la imagen. En este caso, el operador encargado de llevar a cabo las derivadas parciales de segundo orden es el Laplaciano, el cual viene determinado por la expresión indicada en la ecuación 2.5.

$$\nabla^2 = \frac{\delta^2 f}{\delta x} + \frac{\delta^2 f}{\delta y} \tag{2.5}$$

Dentro de los operadores empleados para la detección de bordes y que basan su funcionamiento en las derivadas de segundo orden, se pueden encontrar el operador *Laplaciano de Gaussian (LoG)* así como el operador de bordes *Canny*. Debido a los resultados obtenidos entre ambos, el operador a destacar sin duda es *Canny* ya que sirve para detectar tantos bordes fuertes como débiles presentes en la imagen. Para ello, el proceso a seguir en este operador consiste en realizar un alisado previo en la imagen de tal forma que el ruido posible existente en la función sea eliminado lo máximo posible pero sin verse afectas las características propias de los bordes. Una vez realizada esta tarea, el proceso continúa con la búsqueda de los bordes de la imagen y el correspondiente umbral que permita determinar la separación entre las diferentes zonas a segmentar.

Para concluir con la sección y a modo de resumen, en la figura 2.6 se pueden ver los resultados obtenidos en base al operador aplicado en cada uno de los casos. Observándose como los mejores resultados se obtienen tras aplicar *Canny* sobre la imagen original.

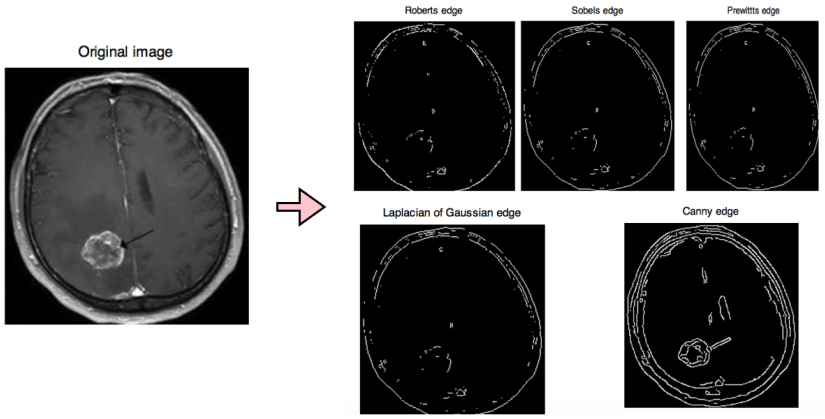


Figura 2.6: Resultados obtenidos en base al operador aplicado [1]

2.2.4 Métodos basados en agrupamientos

Los métodos de agrupamiento o también conocidos como *Algoritmos de Clustering* basan sus principios en la unión de los datos de entrada en diferentes subconjuntos conocidos como *clusters*. De esta forma, el empleo de algoritmos de este tipo en segmentación de imágenes implica una agrupación de píxeles perteneciente a la imagen de entrada en diferentes *clusters*, de tal forma que los elementos que forman cada uno de los *clusters* se caractericen por tener patrones o propiedades semejantes que les hacen a la vez diferentes del resto de píxeles que estarían contenidos en otros *clusters*.

Los métodos de agrupamientos se caracterizan por llevar a cabo las divisiones de los *clusters* acorde a criterios tales como la distancia, empleando para ello funciones distancia como la euclídea o realizando una división en base a la propia similitud, haciendo que se clasifiquen dentro de la categoría de algoritmos de segmentación no supervisados donde los datos de entrada son tratados como un conjunto de variables aleatorias, a diferencia de los algoritmos supervisados, en los que se parte de un conocimiento a priori.

De esta forma, los métodos basados en agrupamiento de datos en *clusters* pueden ser divididos en dos tipos: en el primer grupo se lleva a cabo un agrupamiento jerárquico en el cual se realizan tanto división de *cluster* como posibles unificaciones entre ellos. En el segundo grupo, conocido como agrupamiento no jerárquico, se fija el número de *clusters* que van a ser necesarios, empleando posteriormente criterios de distancia para llevar a cabo la decisión de pertenencia a cada *cluster*.

En cuanto al uso que se da a este tipo de algoritmos basados en el agrupamiento, es importante mencionar que es muy empleado en la minería de datos, bioinformática así como procesamiento de imágenes, campo que compete al estudio que se va a llevar a cabo a lo largo de este documento.

Debido a la importancia y el amplio uso que han tenido los algoritmos de *clustering* a lo largo de la literatura y que este concepto no es tema principal de este estudio, en esta sección se desarrollarán los algoritmos de agrupamiento más conocidos, los cuales estarían comprendidos dentro del tipo *no jerárquicos*.

2.2.4.1 Basados en el error cuadrático

Todos los métodos basado en el cálculo del error cuadrático tienen como objetivo minimizar, hasta obtener el resultado deseado, la función de error cuadrado expresada a continuación en la ecuación 2.6. [3]

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \parallel x_i^j - c_j \parallel^2 \tag{2.6}$$

Donde $\parallel x_i^j - c_j \parallel^2$ representa la media de la distancia entre un punto cualquiera del conjunto de datos, x_i^j y el centro del cluster c_j.

Sin duda, uno de los algoritmos más utilizados dentro de los basados en el error cuadrático es *k-means*. Como ya se ha mencionado anteriormente, este tipo de algoritmos forma parte de los métodos de agrupamiento o *clustering* del tipo no jerárquicos, donde el número de los k *clusters* son definidos desde el inicio. En cuanto a la asignación de cada uno de los datos que forman el conjunto de entrada que se pretende segmentar, éstos son asignados en base a la distancia euclídea hasta el *cluster* más cercano. Otra de las características a destacar de este algoritmo es que hace uso de la media estadística para calcular nuevos *clusters*, obteniendo así el centroide del nuevo *cluster* mediante la media de todos los elementos asignados a él.

De esta forma, el procedimiento a seguir consistiría en asignar de forma previa cada uno de los centroides de los k *clusters* a utilizar de tal forma que se pueda calcular la distancia desde cada punto hasta cada uno de dichos centroides establecidos. Una vez calculadas esas distancias, se asigna cada punto al *cluster* cuya distancia entre su centroide y el punto como tal, sea menor. Una vez que se tiene establecido el reparto de puntos a lo largo de los diferentes cluters que se han fijado, se ajustan los centroides para que se encuentren en el centro del conjunto de datos que pertenecen a ese *cluster*. Una vez fijados los nuevos centroides, el algoritmo se vuelve iterativo y se produce de nuevo el cálculo de distancias entre los puntos y los centroides. El proceso finalizará cuando no se produzca ninguna variación en la posición de los centroides y por lo tanto, cuando el algoritmo haya convergido.

Para simplificar el funcionamiento del algoritmo *k-means*, en la figura 2.7 se puede observar un ejemplo de cómo quedarían divididos unos datos de entrada cualesquiera en base a un valor de $k = 3$.

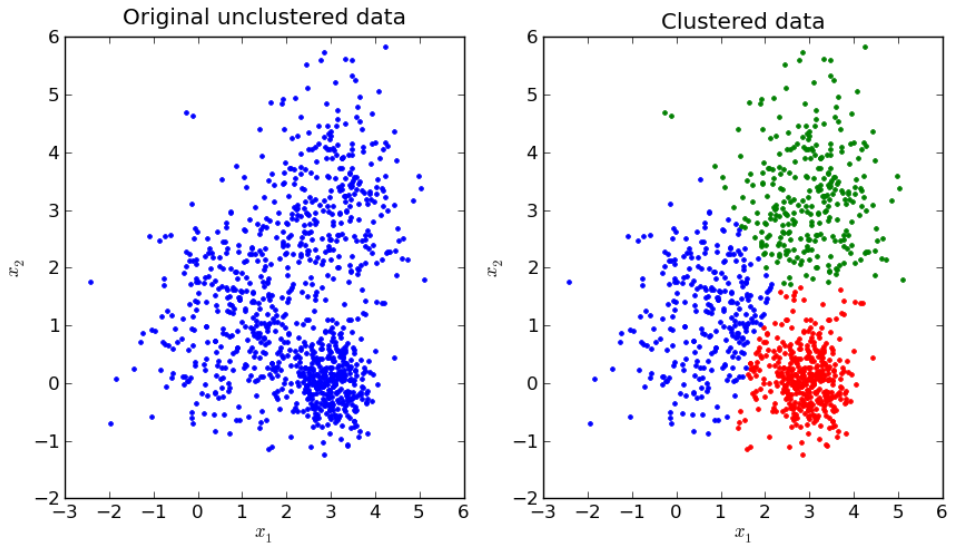


Figura 2.7: Ejemplo de $k-means$ con $k=3$

A la hora de citar las ventajas y las desventajas que este método presenta, es importante destacar el hecho de que, aun siendo un método antiguo y poco robusto, es uno de los más utilizados a la hora de llevar a cabo una segmentación de unos datos de entrada cualesquiera. Además, en muchas ocasiones suele ser utilizado como un método de preprocesamiento.

En cuanto a las desventajas que este algoritmo presenta, se puede destacar el hecho de que no es apto para datos categóricos, como los que se dispone en este estudio. También hay que mencionar el hecho de que no siempre se puede garantizar una convergencia absoluta del algoritmo, lo que implica que no se llegue a encontrar el valor más óptimo en cuanto a la asignación de cada uno de los datos en los diferentes *clusters* de entrada. Dicha convergencia se verá afectada también por la posición inicial de los centroides, la cuál puede hacer que los resultados varíen en gran medida.

2.2.4.2 Basados en grafos

Los métodos basados en el corte de grafos o comunmente conocidos como *Graph Cut* se pueden incorporar dentro de la clasificación de algoritmos de agrupamiento, ya que su objetivo es separar diferentes regiones a fin de llevar a cabo la segmentación de la imagen dada. Este tipo de algoritmo basa sus principios de funcionamiento en la minimización de una función de energía que se puede resolver mediante técnicas utilizadas en teoría de grafos. La combinación del flujo máximo de un grafo da como resultado el corte mínimo que se tiene que hacer en dicho grafo para que este corte sea el más pequeño posible. El corte del grafo permite realizar la segmentación de manera eficiente y óptima.

De esta forma, el procedimiento seguido se basa en la asociación de los píxeles que forman la imagen con cada uno de los nodos de un grafo. Por su parte, los arcos o uniones entre cada uno de los píxeles vecinos se encargan de indicar la similitud que hay entre ellos. Además existen dos vértices claramente diferenciados, S (Fuente) y T (Pozo), que representan las semillas de inicialización establecida al comienzo del algoritmo.

La imagen es modelada de esta forma mediante una red de flujo, donde los diferentes píxeles se asocian con los nodos de la imagen de tal manera que el flujo máximo va aumentando progresivamente desde la fuente S hasta el pozo, T a lo largo del grafo, G. Al final se obtendrá un valor de flujo máximo que satura el gráfico, correspondiendo este valor con el punto por el que se tiene que llevar a cabo el corte que representa el costo mínimo y obtener así una segmentación óptima.

De esta forma, el corte mínimo realizado representa una partición de la imagen en dos regiones en las cuales, el pozo S formará parte de la región C_1 y el pozo, T, de la región C_2. En la figura 2.8 se puede mostrar un ejemplo de posibles cortes en un determinado grafo, G:

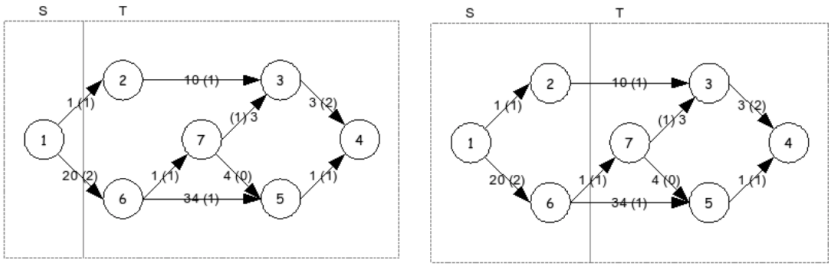


Figura 2.8: Ejemplos de cortes en un grafo

En [1, 24] se profundizan los métodos de este tipo.

Para mostrar otro ejemplo de como se llevaría a cabo una segmentación en una imagen bidimensional en la cual se quiere diferenciar entre objeto y fondo, se muestra la figura 2.9, obtenida de [24]

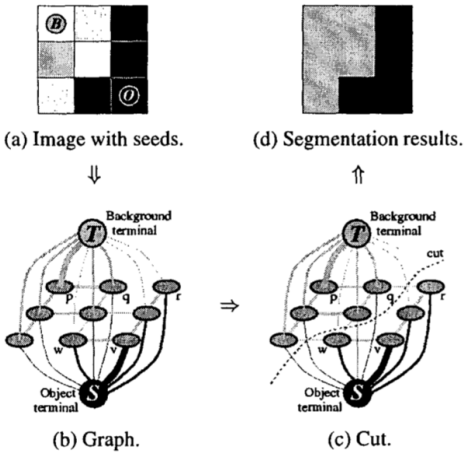


Figura 2.9: Ejemplos de corte en una imagen 2D

Los métodos basados en cortes de grafos son intensivos en memoria, dificultando por lo tanto el procesamiento de imágenes de gran dimensión donde se requiere de gran espacio para conseguir una correcta segmentación. La otra desventaja es la ausencia de métodos sistemáticos para definir las funciones de pesado del grafo que permiten separar correctamente dos clases.

2.2.4.3 Basados en *fuzzy* clustering

Dentro de los métodos de *clustering* se encuentra también el conocido Fuzzy c-means (FCM). Este método de segmentación sigue una idea muy parecida a *k-means* y la diferencia principal reside en el hecho de que en este caso los puntos del conjunto de datos pueden formar parte de varios *clusters* y no necesariamente de uno solo como ocurre con *k-means*. Para ello, el algoritmo *fuzzy* tiene como objetivo la minimización de la función representada en la ecuación 2.7. [3]

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ij}^{m} \parallel x_i - c_j \parallel^2, 1 \leq m < \infty \tag{2.7}$$

Donde u_{ij} representaría el grado de pertenencia de x_i al *cluster* j. Por su parte, c_j representa el centro de dimensión d del *cluster*. De tal forma que al llevar a cabo la optimización de la función hasta la búsqueda del mínimo, lo que se actualiza de forma iterativa es el valor de pertenencia u_{ij} así como los centros de cada *cluster* c_j.

De esta forma, el procedimiento consiste en crear una matriz de pertenencia en la que se indicaría en una escala del 0 al 1 el porcentaje que tiene cada dato de pertenecer a un *cluster* u otro siendo, en este caso, el 1 el correspondiente al 100 % y el 0 con el 0 % de probabilidad. Como bien se ha mencionado, este caso solo se daría con un agrupamiento *k-means* en el que habría un 1 en el *cluster* al que perteneciera el dato y un 0 en el resto. Por el contrario, con un agrupamiento FCM, se podría obtener una matriz de pertenencia como la mostrada en la figura 2.10. En esta se observa la matriz de pertenencia de *k-means* y *Fuzzy c-means* para un conjunto de n datos de entrada.

$$U_{MC} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ .. & .. \\ 0 & 1 \end{bmatrix} \qquad U_{MC} = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \\ 0.6 & 0.4 \\ .. & .. \\ 0.9 & 0.1 \end{bmatrix}$$

Figura 2.10: Matrices de pertenencia para $k = 2$ [3]

De esta forma, para visualizar de una forma práctica los resultados proporcionados por la figura 2.10, supongamos un ejemplo en el que se parte de un conjunto de n datos, donde el objetivo consistiría en dividir esos n datos en dos clústeres diferentes, $k = 2$. De esta forma, partiendo de los datos unidimensionales mostrados en la figura 2.11, los resultados serán diferentes según el algoritmo empleado.

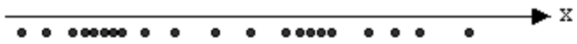


Figura 2.11: Datos de entrada [3]

Partiendo del algoritmo *k-means*, en el que los datos de entrada solo pueden formar parte de un *cluster* u otro, la función de clasificación que corresponde a la función de pertenencia de la figura 2.10 sería la mostrada en la figura 2.12.

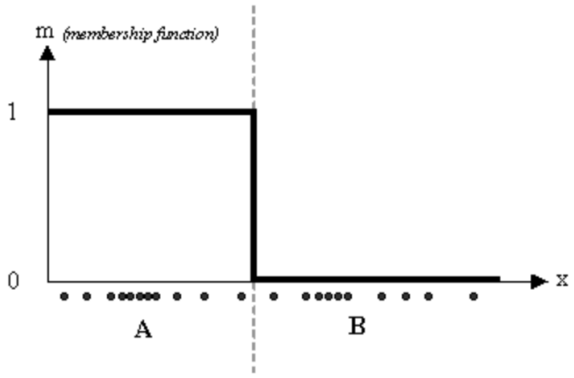


Figura 2.12: Función de pertenencia para $k - means$ [3]

Por su parte, si en lugar de utilizar *k-means* como algoritmo de agrupamiento, se decide emplear una clasificación *Fuzzy c-means*, los resultados obtenidos serán los mostrados a continuación en la figura

2.13. Es ésta se observa como aquellos puntos que no forman parte, de forma clara, de uno de los dos *clusters* disponibles, estarían representados con una función de pertenencia cuya caída desde 1 hasta 0 se realizaría de una forma más suave.

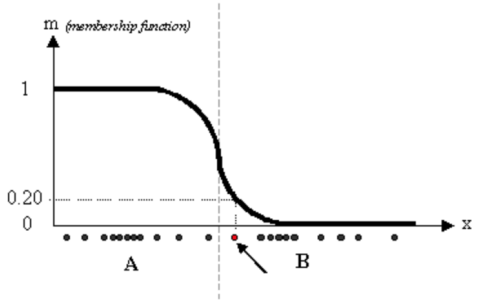


Figura 2.13: Función de pertenencia para FCM [3]

En el caso de que se aumentara el tamaño de k hasta tres *clusters* diferentes, los resultados que se podrían obtener con el algoritmo *Fuzzy c-means* podrían ser de la forma mostrada en la figura 2.14.

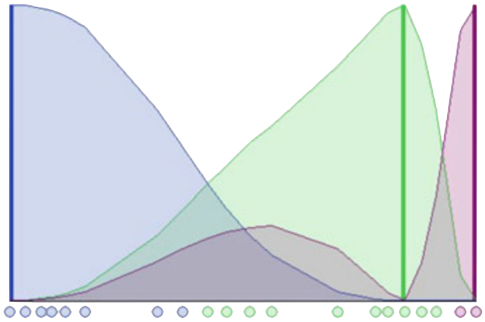


Figura 2.14: Función de pertenencia para FCM con $k = 3$ [3]

2.3 *Deep Learning* aplicado a la segmentación semántica

Desde finales del siglo XX, las redes neuronales artificiales o también conocidas como Artificial Neural Network (ANN), se han ido abriendo paso poco a poco en el mundo de la visión. En la actualidad han alcanzado una gran repercusión en el mundo científico, y en la visión artificial en particular, gracias a las técnicas modernas de *deep learning*. Estas han permitido resolver con asombrosa precisión numerosos problemas que se consideraban abiertos hasta ese momento. Todo ello ha ido acompañado de una auténtica revolución tecnológica en términos de la potencia de cálculo disponible a través de sistemas basados en procesadores gráficos o GPU.

2.3.1 Fundamentos de las redes neuronales

El funcionamiento de una neurona clásica se muestra en la figura 2.15. Toma un vector de entrada y realiza la suma ponderada de las componentes de dicho vector, añadiendo un *offset* al resultado. El resultado escalar de dicha suma se introduce por una función no lineal o función de activación. Los parámetros de la neurona son los pesos de la suma ponderada y el *offset*.

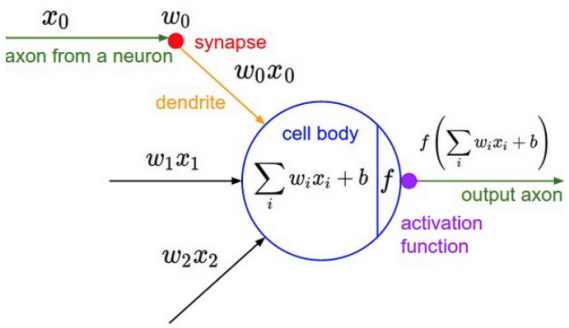


Figura 2.15: Esquema de funcionamiento de una neurona artificial

Existen diferentes tipos de función de activación como *Sigmoid, Tanh, ReLU, Selu, softmax*, pero sin duda la más común hoy en día dentro de las redes neuronales en la activación ReLU. En la figura 2.16 se pueden observar la forma de algunas de las activaciones más comunes.

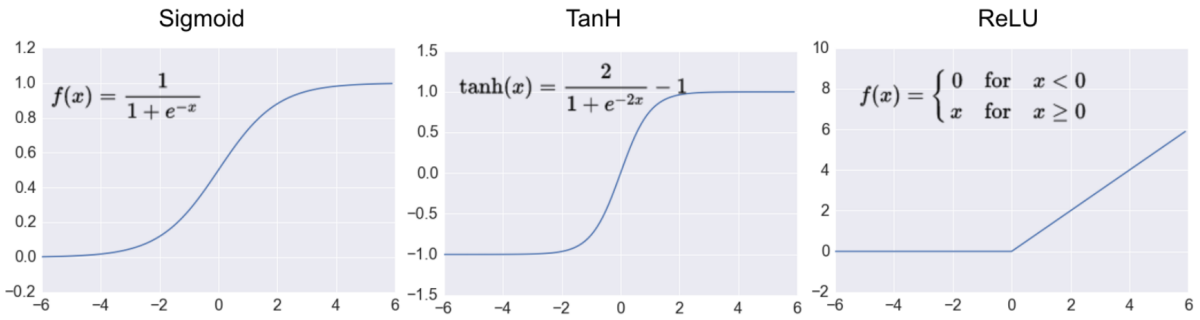


Figura 2.16: Ejemplos de activaciones [4]

Una red neuronal se forma mediante la conexión de múltiples neuronas, todas ellas en diferentes niveles o capas con un número determinado de neuronas por capa. De esta forma, se pueden diferenciar tres tipos de capas en base a la función que cada una de ella desempeñe:

- **Capa de entrada**: encargada de recibir la información proveniente del exterior, información que se utilizará para llevar a cabo la resolución del problema.

- **Capas ocultas**: se encuentran localizadas entre la capa de entrada y la de salida. No existe un número fijo de estas capas en una red neuronal por lo que pueden ir desde ninguna capa oculta hasta un número elevado de ellas. Se encuentran formadas por diferentes neuronas todas ellas interconectadas entre sí lo que determinará la topología propia de la red. Este tipo de capas se caracterizan por no tener un contacto directo con el exterior y estar aisladas. En estas capas es básicamente donde se desempeña el mayor proceso de cómputo.

- **Capa de salida**: es la encargada de transferir la información resultado al exterior una vez que ésta ha sido procesada por las capas anteriores.

A continuación, en la siguiente figura se puede observar un posible esquema de una red neuronal donde estarían localizadas los tres tipos de capas mencionadas:

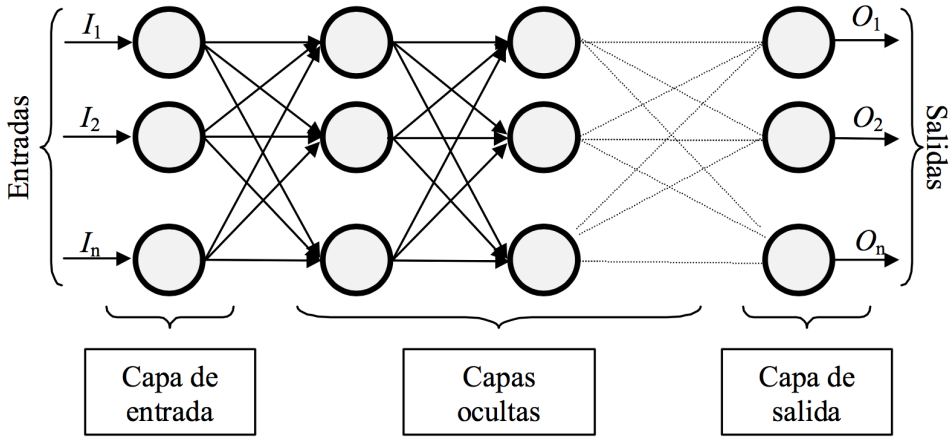


Figura 2.17: Esquema de red [5]

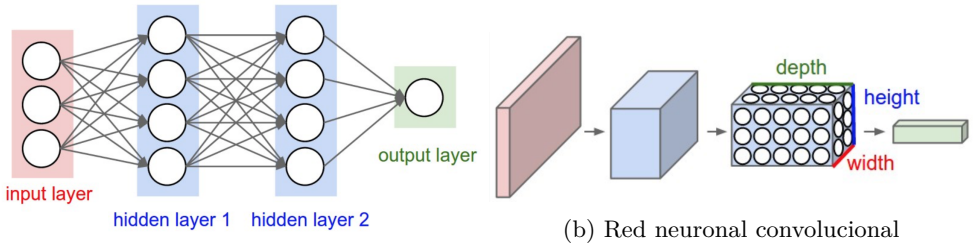
Como bien se ha comentado, cada una de las capas que forman la red neuronal pueden estar formadas por un conjunto de neuronas. El número más adecuado para cada caso no es una decisión trivial ya que la combinación de cientos o millones de ellas puede hacer que sea más sencillo resolver problemas mas complejos pero también conlleva el hecho de necesitar más tiempo de procesamiento para obtener dicha solución.

Por otra parte, las neuronas se pueden dividir en varios tipos en base al valor que puedan tomar: binarias, reales, etc.

Una vez definidas las funciones más importantes que se desarrollan en las ANN, es importante mencionar el concepto de entrenamiento, el cual se citó anteriormente. El entrenamiento de una red neuronal artificial se basa principalmente en el aprendizaje de los pesos de la propia red. Para ello, es necesario definir el conjunto de datos de entrenamiento, con el cual la red llevará a cabo ese aprendizaje de los pesos. Este conjunto de datos puede ser de cualquier tipo, como imágenes o sonido, pero siempre deberá ir acompañado de la correspondiente salida o etiqueta asociada a ese dato de entrada. De esta forma la red pueda llevar a cabo la asociación entre las salidas proporcionadas y los pesos de dicha red. Además de los datos de entrada, es de vital importancia proporcionar a la red de una función de pérdidas que mide mediante un número real el desempeño de la red para predecir las salidas de entrenamiento a partir de las entradas correspondientes. De esta forma, una vez que se dispone de estos dos elementos, se lleva a cabo el entrenamiento de la red [4].

2.4 CNN: Redes neuronales convolucionales

Las *convolutional neural networks* Redes Neuronales Convolucionales (CNN) o *ConvNets* han adquirido en los últimos años una gran relevancia en vision artificial [7]. Estas redes, cuyo origen se data en 1989, difieren de las redes clásicas en la inclusión de capas de un tipo de neurona particular cuya función matemática se basa en la convolucion discreta con un conjunto de filtros de respuesta finita. Las operaciones de convolución se adaptan especialmente bien para el procesamiento de señales con diferentes dimensiones espaciales o temporales, como pueden ser las imágenes o las señales de audio. Además, y en comparación con las redes tradicionales, el número de paramétros que requieren las capas convoluciones no crece con las dimensiones espaciales de la imagen. La contención en el número de paramétros y las características de la convolución contribuyen a la construcción de redes neuronales profundas que operan fácilmente con imágenes.



(a) Red neuronal regular

(b) Red neuronal convolucional

Figura 2.18: Arquitecturas de una red neuronal regular vs una red convolucional [6]

A continuación, en la siguiente figura, se puede observar un ejemplo del cambio producido en cuanto a arquitectura se trata entre las redes regulares y las convolucionales:

Como bien se puede observar en la figura 2.18b, la distribución de las capas con respecto a las de las redes regulares sufre algunas variaciones ya que básicamente, en este caso, las redes convolucionales estarían formadas principalmente por tres tipos de capas diferentes: capa convolucional, capa de *pooling* y las totalmente conectadas o también conocidas *Fully-Connected*, las cuales se detallarán a continuación.

2.4.1 Principales capas de las ConvNets

Como bien se muestran en [6], las redes neuronales convolucionales se caracterizan por el empleo de tres tipos principales de capas: las capas convolucionales, las de *pooling* así como las *fully-connected*.

En la figura 2.19 se muestra una estructura típica de red CNN donde se alternan los tipos de capas anteriormente descritos.

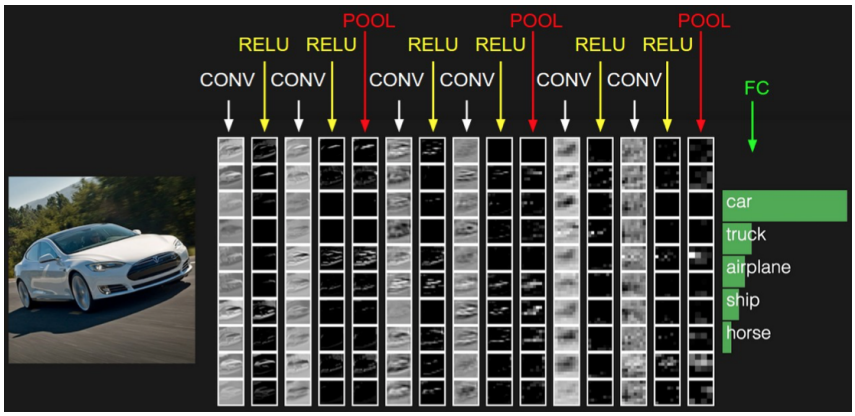


Figura 2.19: Ejemplos de posible estrucutra de red [6]

Se describe a continuación cada una de las capas que componen una red CNN.

2.4.1.1 Capa Convolucional

Sin duda, esta capa es la fundamental dentro de las redes neuronales convolucionales. En estas capas la entrada se compone de un tensor donde se distinguen dimensiones espaciales (alto y ancho en una imagen) y la dimensión de profundidad (canales de color). Los filtros que se definen en esas capas constan de *kernels* de un tamaño generalmente inferior a la imagen de entrada y coinciden con la entrada en la dimensión de profundidad. Por simplicidad asumimos en este trabajo que las entradas de las capas convolucionales son de tipo imagen, con dos dimensiones espaciales y profundidad de color.

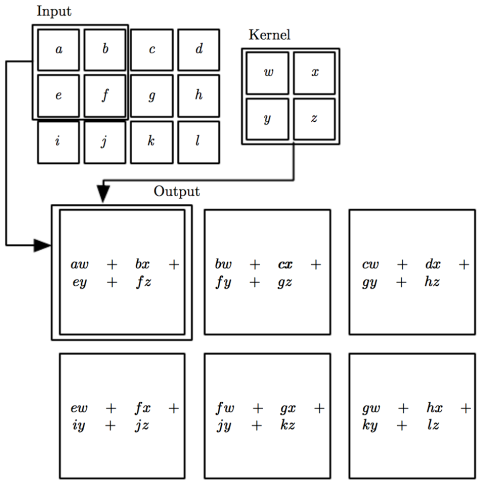


Figura 2.20: Ejemplos de convolución [7]

Así en cada capa convolucional se dispondrá de un conjunto de filtros con unas dimensiones determinadas que se encargará de producir cada uno de ellos un mapa de activación de la imagen, obteniendo a la salida un tensor con tanta profundidad como filtros existan en esa capa.

Dada una imagen I de entrada, representada por una matriz bidimensional y un *kernel* K, la convolución se expresa mediante la siguiente ecuación:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.8}$$

En la figura 2.20 se puede ver cómo se produce la convolución entre una matriz bidimensional de entrada con un *kernel* determinado.Esto origina una salida con unas dimensiones diferentes a las de la imagen de entrada, como bien se puede ver en la ilustración indicada. Es destacable comentar que en este caso el desplazamiento del filtro a lo largo de la superficie de la imagen se hace en base a un parámetro denominado *stride* y que indica el desplazamiento en las dimensiones de la imagen que realiza el filtro durante la convolución.

De este modo, mediante el tamaño del filtro, el número de filtros y el *stride* se obtienen diferentes tamaños de salida de las capas convolucionales.

Las capas convolucionales poseen un número de parámetros inferior a los requeridos por las neuronas clásicas para la mismas dimensiones de entrada y salida. Además, gracias a la convolución permiten aprender características de la imagen con independencia de la posición en la imagen donde se manifiesten. Esta idea surge del hecho de que si una característica es necesaria para poder llevar a cabo el cálculo de una determinada posición implica que dicha característica también puede ser útil para calcular una posición diferente. Este razonamiento resulta ser coherente ya que indica que si la detección de un borde es importante en alguna ubicación concreta de la imagen, conlleva que dicha detección del borde debería ser utilizado en otra parte de la imagen, por lo que una vez aprendido el proceso de detección de bordes, no sería necesario volver a realizar dicho aprendizaje, sino que con la compartición de parámetros quedaría solventado.

Por último, para finalizar con las capas convolucionales, es importante hacer una breve mención acerca de los hiperparámetros más importantes dentro de este tipo de capas y los cuales se encargarán, por lo tanto, de determinar el volumen de salida. Estos hiperparámetros son: ***la profundidad, stride y el relleno con ceros*** o también conocido como *zero-padding*

- ***Profundidad o depth:*** En este caso el *depth* de un volumen de salida representa la cantidad de filtros utilizados en esa capa para la detección de diferentes elementos como pueden ser bordes o manchas de color. De esta forma, un conjunto de neuronas se encargarán de examinar una zona concreta del volumen de entrada para poder llevar a cabo la detección de esos elementos importantes. La profundidad de la capa o el depth es un hiperparámetro que quedaría determinado mediante la letra F.
- ***Stride:*** Este término ya ha sido mencionado con anterioridad en esta misma sección y como bien se ha detallado anteriormente, este hiperparámetro se encarga de especificar el desplazamiento que tiene que hacer el filtro empleado a lo largo de la superficie examinada. De esta forma, si el stride es de uno $S = 1$, entonces esto implica que el filtro realiza desplazamientos de un píxel en un pixel. En su caso, si se tratara de $S = 2$, se produciría un salto de píxel hasta la siguiente posición del filtro. Valores de *stride* superior a $S = 2$ son poco comunes.
- ***Relleno con ceros o zero-padding:*** Este hiperparámetro es muy utilizado cuando se desea rellenar los alrededores de la imagen de entrada con ceros por los bordes. Esta táctica permite mantener que la salida de la capa sea de las mismas dimensiones que la entrada. En este caso, este hiperparámetro quedaría determinado por la letra P

De esta forma, sabiendo el volumen que ocupa la imagen de entrada y el valor de estos tres hiperparámetros, se podría conocer el tamaño que tendría la salida de dicha capa con solo aplicar la siguiente ecuación:

$$W_{out} = \frac{W_{in} - F + 2P}{S} + 1 \tag{2.9}$$

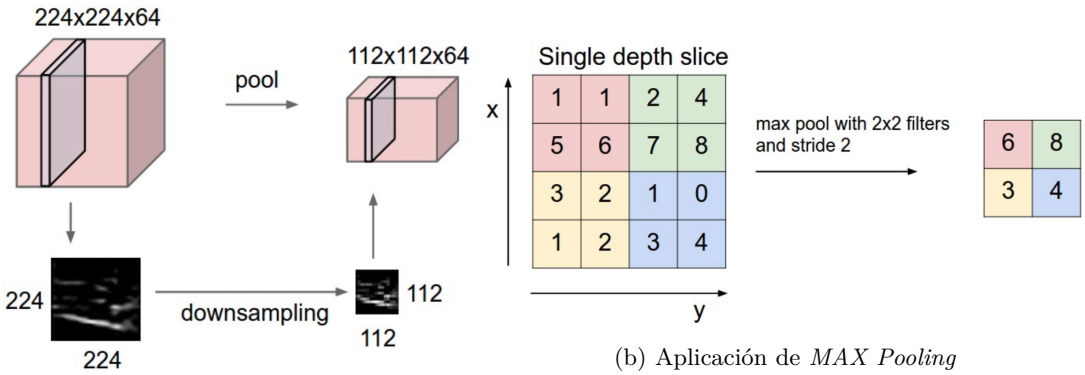
En la cual, W_{in} y W_{out} representarían respectivamente, la superficie del volumen a la entrada y salida de la capa.

2.4.1.2 Capa *Pooling*

Es muy común ver en la mayoría de arquitecturas de redes convolucionales una capa de *pooling* insertada entre varias capas convolucionales sucesivas con el objetivo primordial de disminuir de forma progresiva el tamaño espacial de la imagen, de tal forma que el número de parámetros empleados se vea reducido de la misma forma, evitando así manejar unas cantidades muy elevadas de parámetros que dificultarían el proceso de cálculo.

De esta forma, las capas de *pooling* pueden llegar a utilizar varios tipos de operaciones, *máximo, media o la normal L2*, pero si duda la más empleada hoy en día es la operación máxima la cual se encarga de coger un número determinado de valores que representarían los máximos de una sección determinada. Para establecer la sección donde coger esos valores máximos, se suele emplear un filtro de pequeñas dimensiones en el que suele ser muy común regiones de 2 x 2 el cual se iría desplazando a lo largo de la superficie en base a una valor de *stride* o desplazamiento fijado también de forma previa. De esta manera, la superficie es analizada con estos pequeños filtros para establecer, dentro de cada región, los máximos valores y reducir así la superficie total. Para comprender mejor el proceso que se desempeña en las capas de *pooling*, en la siguiente figura se puede observar como se produce la ya comentada reducción de tamaño y como, en la figura 2.21b se puede apreciar la elección del valor máximo para esa reducción

Por último, comentar que este tipo de capas requieren de dos parámetros para llevar a cabo de forma correcta su función, uno sería las dimensiones espaciales de la imagen F (alto y ancho) y el otro correspondería con el *stride* S. De esta forma, el *stride* indicará el desplazamiento del filtro así como la dirección. Por ejemplo, si $S = 2$ entonces la región del filtro se desplazará dos píxeles a la derecha para analizar la siguiente región de la superficie original y determinar así el valor máximo en esa parte.



(a) Redución en las dimensiones espaciales

(b) Aplicación de *MAX Pooling*

Figura 2.21: Ejemplo de funcionamiento de una capa de *pooling* [6]

2.4.1.3 Capa *Fully-Connected*

Este tipo de capas realizan una operación similar a las capas de neuronas clásicas. Se caracterizan principalmente por disponer de conexiones a todas y cada una de las activaciones de la capa anterior. De esta forma, se puede decir que la diferencia principal con las capas convolucionales es que éstas se encuentran conectadas a una pequeña región de la entrada a diferencia de las *fully-connected* y que además comparten parámetros entre ellas.

Existen otros tipos de capas que se emplean a la hora de crear la estructura de una red y que por lo tanto, merecen una mención, pero al no tratarse de las principales capas que se suelen implementar en las redes convolucionales, serán mencionada en el capítulo siguiente, 3 en el cual se detallará la estructura de red propuesta y por lo tanto, el motivo de la elección de cada una de las capas que la conforman.

2.4.2 Principales arquitecturas basadas en CNN

Se enumeran y describen brevemente algunas arquitecturas de redes CNN relevantes por su versatilidad y el impacto reciente que han tenido en la comunidad científica:

- ***LeNet:*** Es el nombre que recibió la primera red neuronal basada en convoluciones. Fue creada en torno a 1990 de la mano de Yann LeCun y empleada para leer códigos digitales entre otras cosas. Esta red supuso el inicio de las redes convolucionales.

- ***AlexNet:*** Desarrollada por Alex Krizhevsky, Ilya Sutskever y Geoff Hinton tuvo su origen más de dos décadas después de LeNet, en la cual se basó significativamente a la hora de llevar a cabo la creación de la estructura de red. A diferencia de LeNet, contenía capas convolucionales de mayor tamaño que su sucesora así como más profundas.

- ***ZF Net:*** Con solo un año de diferencia, ZF Net fue creada por Matthew Zeiler y Rob Fergus como mejora de LeNet, donde en este caso se había llevado a cabo un ajuste de los hiperparámetros que formaban la red.

- ***GoogLeNet:*** Ganadora en 2014 del *ILSVRC* y creada por e Szegedy et al. Su principal ventaja reside en la reducción en el número de parámetros de varias formas posibles. Una fue incorporando un módulo de inicio además de emplear capas de *pooling* con la operación de promediado, sustituyendo de esta forma las capas totalmente conectadas situadas en la parte final de la red.

- ***VGGNet:*** Ese mismo año en el *ILSVRC*, el segundo puesto fue ocupado por VGGNet, diseñada por Karen Simonyan y Andrew Zisserman. Esta red se caracteriza básicamente por un conjunto de 16 capas tanto convolucionales como *fully-connected* en las cuales las convoluciones realizadas son siempre del mismo tamaño, $3x3$ así como la región empleada en la capa de *pooling*, la cual era de tamaño $2x2$. La desventaja principal que esta red presenta es la cantidad de parámetros que maneja hasta la obtención del resultado, en torno a los 140 millones.

Tras conocer algunas de las arquitecturas que han marcado un antes y un después dentro de las redes convolucionales, es importante mencionar que además de las enumeradas anteriormente son muchas las que han sido conocidas dentro del campo de la visión artificial por los avances que han supuesto. Debido a la extensa lista que se podría hacer de todas ellas, no se va a entrar en profundidad por no ser una tema fundamental para este trabajo, pero sí se hará una mención especial a una serie de arquitecturas que han servido de inspiración en alguna parte del desarrollo de este trabajo. Tal es así, que las redes convolucionales que en este caso merecen una especial atención son: ***UNet, SegNet y ResNet*** las cuales son detalladas a continuación:

2.4.2.1 UNet

Sin duda, la UNet [8] está actualmente catalogada como una de las mejores redes convolucionales profundas aplicadas en el campo de la biomedicina, ya que su aplicación estrella se sitúa en la segmentación de imágenes médicas mediante un proceso rápido y preciso. Se trata de una arquitectura relativamente nueva pues sus orígenes datan del 2015 cuando se proclamó como ganadora de dos *challenges* diferentes: *the Grand Challenge for Computer-Automated Detection of Caries in Bitewing Radiography at ISBI* así como *the Cell Tracking Challenge at ISBI* , superando con un gran margen a todos sus adversarios. En la figura 2.22 se puede observar la arquitectura de esta red, la cual será detallada a continuación.

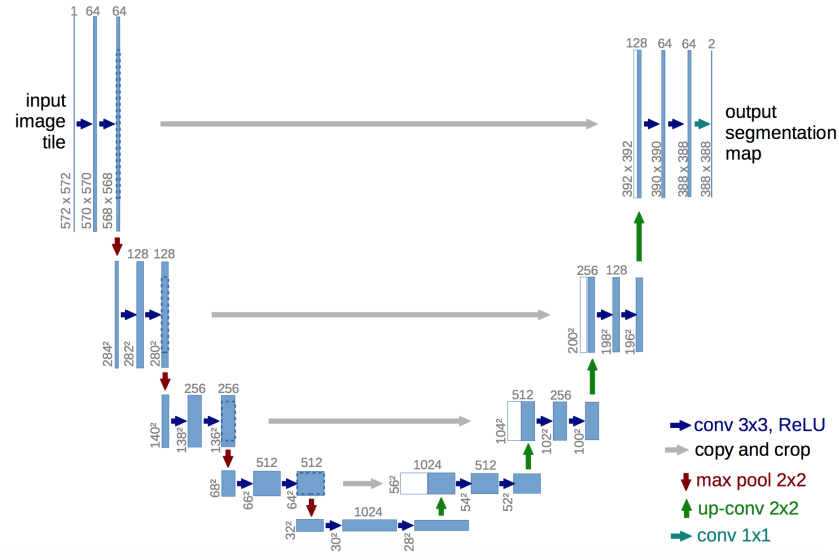


Figura 2.22: Estructura UNet [8]

La arquitectura de la UNet está dividida en dos partes. Una primera parte, perteneciente al lado izquierdo del esquema, se corresponde con el camino de compresión en el cual se mantiene una arquitectura típica de las redes convolucionales. Está formada por la repetición de dos capas convolucionales consecutivas de dimensiones 3 x 3 y sin *zero-padding*, seguidas, cada una de ellas de una activación tipo RELU. Tras estas dos convolucionales se sitúa una capa de *max-pooling* de dimensiones 2 x 2 y *stride* de

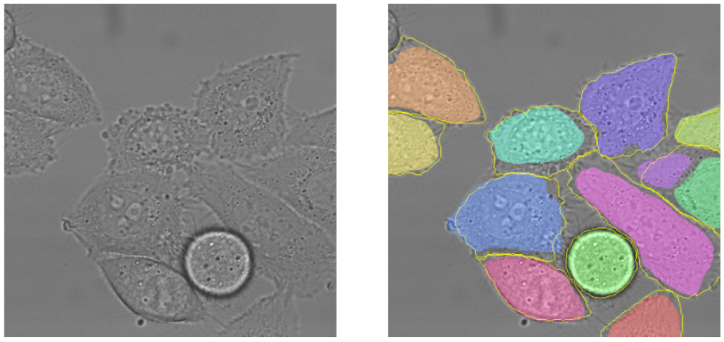


Figura 2.23: Ejemplo de resultados obtenidos con UNet [8]

dos con el objetivo de llevar a cabo la reducción de las dimensiones en el volumen de datos recibido en la entrada.

En lo que respecta a la segunda división de la red, ésta estaría considerada como la parte de expansión en lo que respecta a las dimensiones del volumen recibido. Su estructura consiste en la realización de un muestreo ascendente con convoluciones de 2 x 2, a las cual se incorporaría los mapas de características o *feature maps* heredados de la parte de compresión de la red. Seguida de esta concatenación, se realizarían dos convoluciones idénticas a las de la parte de compresión, es decir, de 3 x 3 y con activación RELU. Finalmente, en la última capa de la red, se llevaría a cabo una convolución de 1 x 1 con la intención de obtener un vector con en el que se encuentren asignadas las características para cada una de las clases segmentadas. De esta forma, con la parte de compresión así como la de expansión, la arquitectura UNet cuenta con un total de 23 capas dispuestas de la forma que se ha indicado.

En cuanto a los resultados que esta arquitectura es capaz de ofrecer, es importante mencionar que alcanza un rendimiento bastante bueno en tareas de segmentación semántica. La estructura de la UNet, donde existe una comunicación directa entre las capas de compresión a las capas de expansión, la hace muy eficiente para entrenamientos con bases de datos de pequeño tamaño y permite limitar la pérdida de información de alta frecuencia. Algunos de los resultados de la UNet se muestran en la figura 2.23, de ello se puede apreciar cómo la separación de las diferentes clases se ha obtenido de una forma bastante satisfactoria.

Añadir, en cuanto a resultados, que en este caso, UNet obtuvo un IoU del 92.03 (%) así como del 77.56 (%) en los dos *dataset* empleados para en el *challenge the Cell Tracking Challenge at ISBI* . Donde la figura 2.23 correspondería al *set* con un IoU más bajo. Aun así, los resultados obtenidos fueron bastantes buenos para lo que en ese momento se estaba consiguiendo dentro del campo de la segmentación semántica en imágenes médicas.

En cuanto al entrenamiento que este tipo de red necesita es relativamente poco, pues los resultados vistos hasta hora fueron obtenidos con un total de 10 horas en una tarjeta NVidia Titan GPU de 6 GB.

2.4.2.2 SegNet

Esta arquitectura tuvo sus orígenes en el 2015, escasos meses después de la aparición de la UNet. Sus creadores, Vijay Badrinarayanan, Alex Kendall y Roberto Cipolla pertenecientes a la Universidad de Cambridge, desarrollaron una red cuya principal tarea se puede situar en el campo de la segmentación semántica en imágenes de tráfico.

La propuesta realizada fue considerada como una idea bastante novedosa dentro del campo de la segmentación pues se encargaba de llevar a cabo esta tarea con una arquitectura formada por dos conjuntos

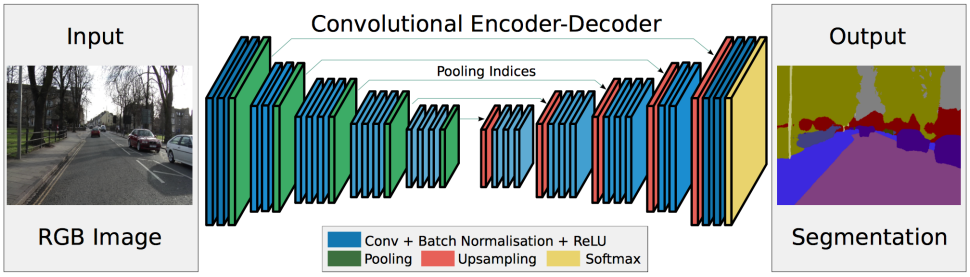


Figura 2.24: Estructura SegNet [9]

de redes neuronales profundas conectadas entre ellas, conociéndose con el nombre de *encoder- decoder* o *codificador - descodificador*. En cuanto a la parte del *encoder*, se trataría de una ligera modificación de la red VGG16 a fin de poder llevar a cabo una extracción de información del objeto. En lo respectivo a la parte del decoder, ésta se encargaría de recibir la información correspondiente y codificarla. A continuación, para facilitar la lectura, se incorpora el esquema correspondiente a la arquitectura de la SegNet, donde se puede apreciar claramente las dos partes significativas comentadas hasta ahora y en las cuales se profundizará a lo largo de este apartado.

En lo que respecta a la entrada, ésta estaría formada por una imagen RGB que tras su paso por toda la red, se obtiene una imagen segmentada en la que se identificaran las posibles clases que puede existir en la imagen de entrada. De esta forma, al tratarse de imágenes relacionadas con el tráfico, las clases que se suelen encontrar son las correspondientes a los vehículos, la carretera, señales, etc. Proporcionando en este caso una ubicación mucho más precisa del objeto en lugar de situarlo mediante un cuadrado delimitador en el que apenas se puede diferenciar los límites de dicha detección.

De esta forma, en la figura 2.24 se puede apreciar en mayor detalle el tipo de imagen empleada para la entrada así como la correspondiente salida que se esperaría. Además, se puede observar en detalle el esquema de la red formado por esa arquitectura tan significativa de *encoder - decorder*.

En cuanto a la arquitectura de la SegNet, es importante mencionar que la parte correspondiente al **encoder** estaría formada por una estructura básica de red neuronal convolucional, en la que las capas más importantes sería la propia capa de convolución. Esta se encargaría de extraer cada una de las características locales de la forma que se ha detallado en la sección 2.4.1.3. Seguida a esta capa, se encontraría un capa de *Batch Normalization*, donde el fin que se persigue es el de normalizar la distribución de los datos de entrenamiento con el objetivo de acelerar el proceso de entrenamiento al situarse todos los datos siempre dentro del mismo rango de valores. Seguida a esta capa, como se puede observar en la figura, se encuentra la capa de activación, en este caso se trata de una activación no lineal, concretamente de una capa RELU.

Por último, posterior a cada uno de los conjuntos de capa convolucional, *Batch Normalization* y activación RELU, se encuentra una capa de *Pooling* cuyo principal objetivo es el de mostrar el mapa de características y propagar la función invariante espacial a la parte correspondiente del *decoder*.

De esta forma, el funcionamiento básico de la red consistiría en que la parte codificadora se encargaría de extraer la información de los diferentes objetos que se encuentran en la imagen, vehículos, carretera, etc; todos ellos representados mediante píxeles a color. Por su parte, el decodificador recibiría esa información, y se encarga de situarla en una imagen de las mismas dimensiones de tal forma que aquellos objetos que pertenezcan a la misma clase se encuentren representados por el mismo color. Identificando así cada uno de los píxeles que forman la imagen con las posibles clases existentes en ella.

Con esto, la tarea que desempeña por lo tanto la parte de **decoder**, se corresponde con la de identificar esa información que ha recibido sobre qué objetos son y la ubicación de ellos, pero añadiendo detalles

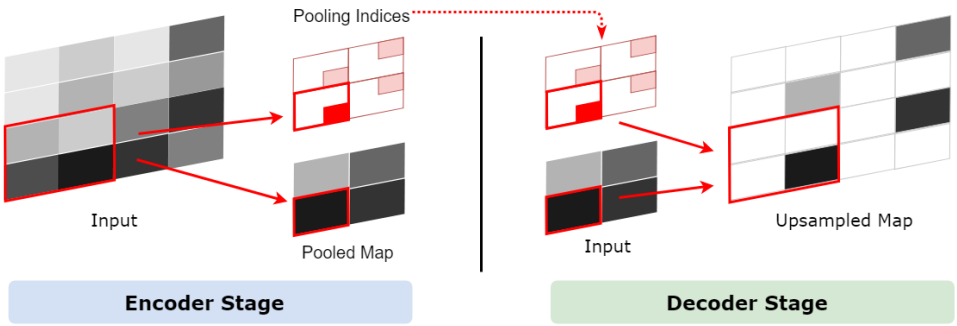


Figura 2.25: Funcionamiento de los índices de *pooling* [10]

geométicos a los mapas de características mediante las correspondientes capas de *Upsampling* para, posteriormente, realizar las convoluciones necesarias, conocidas también como *Deconvoluciones*, para que posteriormente el resultado sea transportado a las capas de *Batch Normalization* y activación RELU; como bien se puede apreciar en a figura 2.24. De esta forma se obtendría la imagen final de las mismas dimensiones que la imagen de entrada y con las clases delimitadas proporcionando así un resultado más suave que el que se podría tener nada más finalizar la parte correspondiente al *encoder*.

Con esto, finalizaría el funcionamiento básico que se lleva a cabo en SegNet, pero como bien se puede observar en la figura 2.24, aparece otro término importante que los creadores denominaron como *Pooling Indices* donde entraría en juego las capas de *Pooling* pertenecientes a la parte del *encoder* así como las capas de *Upsampling* pertenecientes al *decoder*.

La mecánica que se desarrollaría para obtener estos índices que se transportan desde el *encoder* hasta del *decoder* consistiría en la realización de varios pasos. El primero de ellos sería el correspondiente a la ejecución de la capa de *max pooling*, en la cual, como ya se ha detallado en la sección 2.4.1.2 consistiría en determinar el valor máximo de una superficie con unas dimensiones determinadas. En este caso se trataría de una región de $2x2$ con una *stride* de $S = 2$ de tal forma que su ejecución sería similar a la mostrada en la figura 2.21b. Una vez se tiene los índices de las posiciones donde se encuentran los valores máximos, éstos son almacenados en los denominados *Pooling Indices*, los cuales son transmitidos directamente a las capas de *Upsampling* correspondientes a la segunda parte de la red, el decodificador.

De esta forma, una vez que las capas de Upssamplig reciben la información correspondiente a su nivel y contenida en los *Pooling Indices* así como el valor de los máximos obtenidos tras la capa de *pooling* es posible poder asignar cada uno de esos valores en las posiciones correctas como bien se puede observar en la figura 2.26 para, posteriormente llevar a cabo la convolución pertinente.

Así, tras la aparición de esta nueva arquitectura y de las novedades que incorporaba, se llegaron a obtener resultados como los mostrados a continuación. Donde se puede apreciar como la segmentación realizada en diferentes imágenes de tráfico es bastante buena en comparación con los resultados que por aquel entonces se estaban consiguiendo pues según se especifica en el *paper* original [9], se obtuvieron unos valores por encima de las métricas que otras arquitecturas aportaban en cuanto al *mean* - IoU, es decir, media de la intersección sobre la unión, concepto que será detallado en capítulos posteriores, o la exactitud promedio de la clase.

Por último, al tratarse SegNet de una arquitectura que surgió pocos meses después de la UNet y al ser ésta una de las utilizadas como estado del arte para el trabajo que se ha desarrollado, es importante conocer las principales diferencias entre ellas, las cuales consistirían principalmente en el hecho de que UNet está dedicada principalmente a la tarea de segmentación en imágenes médicas donde los resultados que obtiene están por encima de cualquier otra arquitectura pero que sin embargo, cuando se estrapola a otro tipo de imágenes, la segmentación obtenida no llega a ser tan buena. Por su parte, SegNet centra

Figura 2.26: Estructura SegNet [9]

toda su atención en imágenes de tráfico principalmente aunque también ha sido probadas en imágenes de interior obteniendo resultados bastante superiores a otras arquitecturas.

En cuanto a la arquitectura de cada una, la diferencia principal se encuentra en el esquema *encoder-decoder* que proporciona SegNet junto con el empleo de los conocidos índices de *pooling*, ya que en este caso, UNet lo que transmite es el mapa entero de características a sus correspondientes capas de deconvolución, a costa, como es de esperar, de necesitar más memoria para ello.

En lo que respeta a la información relativa a esta arquitectura, se puede encontrar con mayor detalle en el *paper* original citado en [9] así como en [10], referencias que han servido de inspiración para los conocimientos que se han aportado a lo largo de esta sección

Aunque la SegNet fue creada hace relativamente poco, teniendo en cuenta los avances tan significativos que se producen cada poco tiempo, esta red puede considerarse como una arquitectura vieja. ya que hoy en día cada vez son más las arquitecturas nuevas que parten de muchos conocimientos expuestos en las redes convencionales añadiendo novedades que las diferencien del resto. Dentro de la redes destinadas a la segmentación y consideradas como nuevas redes, tiene una mención especial arquitecturas como la **ERFNet** [25, 26], dedicada a la segmentación de imágenes de carácter vial. Dentro de las ventajas que presenta esta red, cabe destacar el hecho de que es ejecutada en tiempo real proporcionando una alta calidad en la segmentación de sus imágenes así como una arquitectura significativa.

2.4.2.3 ResNet

Sin duda, otra de las arquitecturas que ha supuesto una referencia es la denominada ResNet, la cual se caracteriza principalmente por llevar a cabo un aprendizaje residual en redes profundas. Su objetivo no es otro que el de tratar el problema que aparece cuando se lleva a cabo un aumento en la profundidad de las redes, pues dicho aumento debería traer consigo un aumento de la precisión de la red, concepto que no se cumple ya que al producirse un aumento en la profundidad de la red, la señal que surge al final se vuelve muy pequeña en las capas anteriores por lo que tras el aumento, la señal obtenida pierde calidad, lo que hace que el error se vea incrementado a medida que la red va ganando profundidad con el aumento del número de capas. [27]

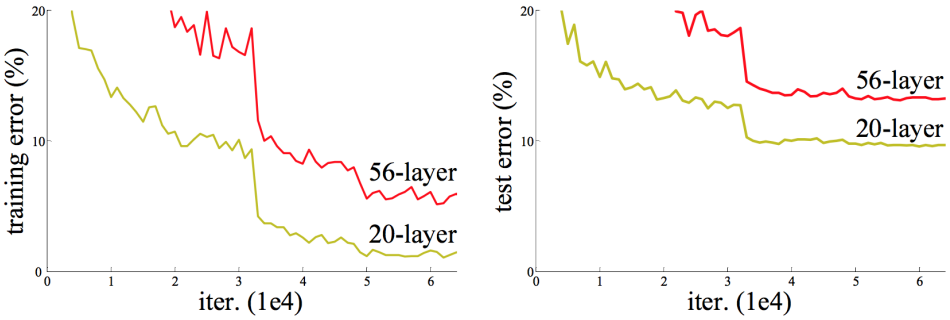


Figura 2.27: Errores obtenidos con el incrementos de capas [9]

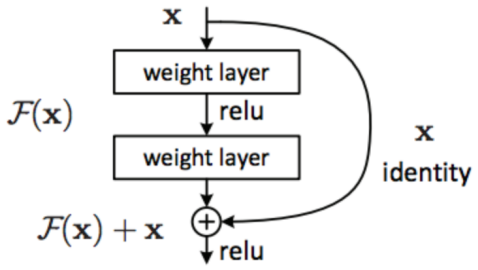


Figura 2.28: Estructura del bloque residual empleado por ResNet [11]

[27] Otro de los problemas que conlleva el incremento en la profundidad de las redes es el realizar una correcta optimización cuando el número de parámetros existentes es bastantes significativo, haciendo de esta forma, que al no poderse optimizar de forma correcta, el error de entrenamiento sea mayor con el incremento de nuevas capas. Esto se puede ver con claridad en una de las gráficas incorporadas por los creadores en el *paper* original [11], la cual se puede observar a continuación.

En la figura 2.27, como bien se puede apreciar, a medida que se aumenta el número de capas, el error obtenido en la fase de entrenamiento va aumentando, haciendo de esta forma que el error obtenido en la fase de *test* sea también mayor. Es entonces en este punto cuando, partiendo con las arquitecturas y conceptos que se conocían, se tenía que llegar a una decisión de compromiso en la que se opte por aumentar o no el número de capas para obtener una mayor profundidad sabiendo que los errores tanto de entrenamiento y de *test* iban a ser mayores haciendo así que la precisión se sature poco a poco y hasta tal punto que llegue a degradarse.

Para evitar este problema, Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun plantean una estructura como solución al problema propuesto. Esta solución consistía en un bloque residual cuya estructura era la mostrada a continuación en la figura 2.28

De esta forma, se cambia el concepto de ajuste y en lugar de esperar que la salida $F(x)$ procedente de la pila de capas se ajuste de forma directa, se produce un residuo procedente de la entrada x de tal forma que a la salida del conjunto de capas no lineales se produzca la suma de dicha salida con el residuo procedente de la entrada, obteniéndose así $F(x) + x$ para posteriormente hacer pasar dicho resultado por el siguiente conjunto de capas.

De esta forma, se ha llegado a observar que el empleo de estos bloques de residuo no solo reduce el error producido sino que además son capaces de producir una disminución en el número de parámetros empleados por la red. Este hecho, se puede apreciar de forma muy clara en uno de los experimentos incorporados en el *paper* original [11]:

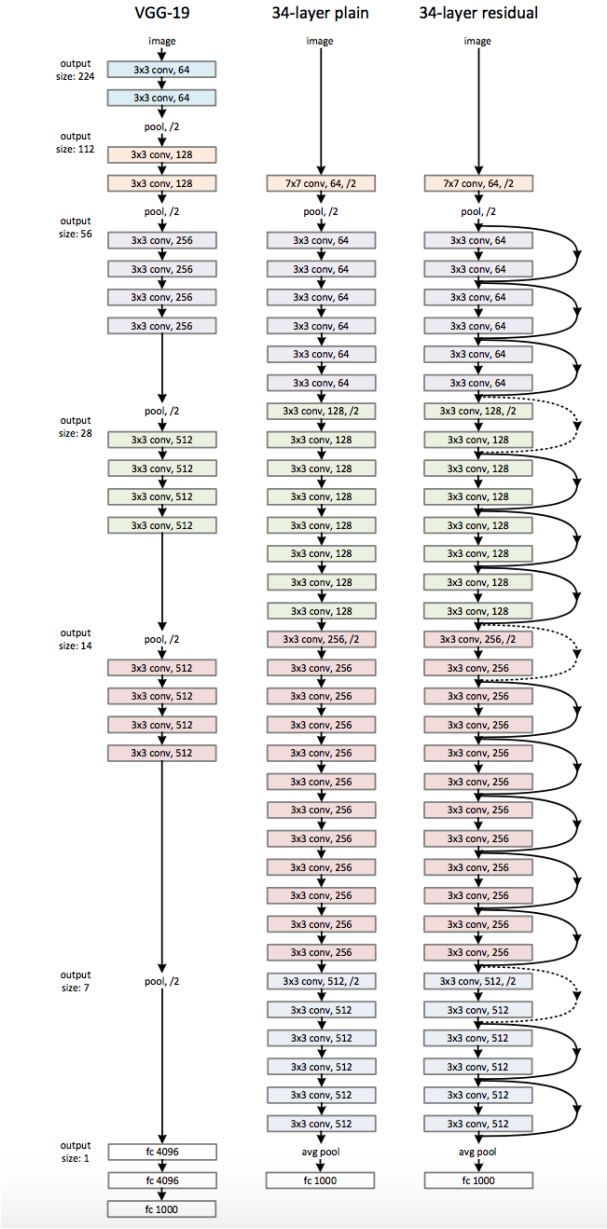


Figura 2.29: Experimento realizado con ResNet [11]

Donde se puede observar que tras llevar a cabo una prueba con la misma base de datos, *ImageNet* en este caso, sobre tres arquitecturas diferentes incluyendo una de ellas los citados bloques residuales, los resultados obtenidos son muy llamativos.

Partiendo de los conocimientos expuestos en [11], la arquitectura de la izquierda representa el modelo de la conocida VGG-19 donde el número de capas que la forman es 19. Por su parte, la arquitectura situada en el centro de la figura, representa una red simple formada por 34 capas. Por último, a la derecha se sitúa la misma red simple, con las mismas 34 capas pero en la cual se han introducido los mencionados bloques residuales.

Esta figura permite ver con claridad cómo se vería modificada una red al introducir los citados bloques de identidad pero en cuanto a disminución del error obtenido no se puede ver a simple vista. Para ello, a lo largo del documento oficial, se incorporan una serie de tablas en las que se demuestra que el error obtenido en validación para determinadas arquitecturas, se ve reducido con la incorporación de estos bloques [11]. De esta forma, tras probar una red simple diseñada para este experimento con un total de 18 capas, el error obtenido en validación es del 27.94 (%) viéndose disminuido hasta el 27.88 (%) con la incorporación de los bloques residuales. Pero sin duda, la muestra más clara de funcionamiento se encuentra en el empleo de una red de profundidad superior formada por 34 capas y con un error superior al caso anterior como era de esperar, 28.54(%). Sin embargo, tras la incorporación de estos bloques, la misma red de 34 capas llega a conseguir un error de validación por debajo del obtenido para el caso de una red simple de 18 capas, y claro está, por debajo de su homóloga sin bloques residuales, bajando este valor hasta el 25.03 (%). Demostrándose así que la incorporación de la idea propuesta funciona gratamente.

Por último, se incorpora además una comparativa práctica en la que se vuelve a demostrar el beneficio del empleo de bloques residuales al confirmarse que cuando más profunda sea la red residual, menor error de validación va a producir. De esta forma, en la siguiente tabla se pueden observar los errores obtenidos tanto para el top-1 como el top-5 para diferentes arquitecturas. Analizadas todas ellas con el mismo set de datos, donde se ha empleado ImageNet para obtener los diferentes errores de validación como bien se puede observar en la tabla 2.1.

modelos	**top-1 err.**	**top-f err.**
VGG-14	28.07	9.33
GoogLeNet	-	9.15
PReLU-net	24.27	7.38
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Tabla 2.1: Resultados comparativos.

De esta forma, partiendo de los buenos resultados obtenidos con el empleo de los bloques residuales frente arquitecturas tan conocidas como *VGG o GoogLeNet*, se ha optado por incorporar parte de sus fundamentos en la propuesta de red que se lleva a cabo en este trabajo y la cual puede verse en detalle en el siguiente capítulo 3.

2.5 Conclusiones

A lo largo de este capítulo se ha podido ir viendo la evolución en los diferentes métodos de segmentación que ha ido habiendo a lo largo de la historia así como las características de cada uno de ellos con sus correspondientes ventajas y desventajas hasta dar paso a la inteligencia artificial, donde el estudio del comportamiento del cerebro y cómo llevar ese funcionamiento a tareas realizadas por un ordenador, ha supuesto un gran avance en el campo de la visión.

De esta forma, se ha hecho un breve repaso sobre los conceptos más importantes de las redes neuronales artificiales o ANN y cómo intentan asemejar el comportamiento de una neurona humana al reproducir de forma muy similar las diferentes tareas que se desempeñan dentro del cerebro humano, haciendo especial hincapié en los conceptos más importantes como lo son la estructura básica que siguen, las posibles capas que puede haber y la función que desempeñan las capas de activación.

Más en profundidad, una vez conocido los principales fundamentos de las redes neuronales, se ha podido estudiar todo lo relacionado con un tipo de éstas en concreto, las redes neuronales convolucionales. Detallando cual es su principio que las hace diferentes a sus predecesoras, las redes neuronales regulares y describiendo en detalle las capas más significativas que suelen emplearse en sus arquitectura como lo son la capa de *pooling*, las *fully-connected* y la más importante de todas, las capas convolucionales.

Posteriormente se ha podido ver un avance cronológico de todas las arquitecturas que han supuesto un momento significativo dentro de este campo, recorriendo algunas de ellas desde LeNet o AlexNet, hasta la conocida VGGNet, para terminar entrando en detalles en tres estructuras de red que han servido de base para el trabajo detallado en este documento, ResNet, SegNet y UNet destacando de ellas la arquitectura que tienen, así como la ventaja que aportan en cada una de las actividades en las que son más apropiadas a utilizar, dentro todas ellas, de la segmentación en imágenes, tarea principal que acontece a este trabajo.

Es por eso, que una vez realizado lo que es considerado como estado del arte, se da paso al siguiente capítulo. En él se lleva a cabo una propuesta de red, definiendo las capas que se van a emplear y el motivo de cada una de ellas así como, todas aquellas características propias de la red propuesta para este trabajo. Dicho modelo se pretende comparar con las dos arquitecturas del estado del arte a fin de observar los diferentes resultados entre cada una de ellas para determinar, de esta forma, cuál sería la arquitectura más adecuada para la tarea que se desea desempeñar.

Capítulo 3

Modelado del problema

No hay que temer a nada en la vida, solo tratar de comprenderlo.

Marie Curie, Cuento de buenas noches para niñas rebeldes

3.1 Introducción

En este capítulo se expondrá la arquitectura de red CNN propuesta para afrontar un problema de segmentación semántica en imágenes de laparoscopia, tal y como se detalló en el capítulo 1. La arquitectura desarrollada ha sido creada a partir de las ideas que algunas de las arquitecturas más conocidas y comentadas en el capítulo anterior.

Por su parte, también se especificarán las diferentes bases de datos que se han empleado para comprobar la funcionalidad de dicha red propuesta, a fin de poder comparar sus resultados con los obtenidos en las redes definidas como estado del arte. Estos resultados serán mostrados posteriormente en el capítulo 4, donde se detallarán en profundidad.

Tras finalizar la descripción detallada de las diferentes bases de datos, se especificarán las diferentes estrategias seguidas a la hora de la realización del entrenamiento de la red, tratando los temas de optimización, *fine tuning*, aumento de datos, etc.

3.2 Propuesta de red

En este trabajo se propone una arquitectura de red como la mostrada en la figura 3.1, donde se pueden observar las diferentes capas que la forman así como la disposición y tamaño de cada una de ellas. Desde este momento, la arquitectura planteada pasará a denominarse como Minimally Invasive Surgery Network (MIS-Net).

La arquitectura propuesta se caracteriza por hacer uso de algunas de las ideas expuestas por SegNet así como por ResNet. Se utiliza una arquitectura *encoder-decoder*, al igual que en SegNet [9]. Las capas convolucionales están inspiradas en bloques residuales propuestos en la ResNet y permiten aumentar la profundidad de la red sin afectar negativamente a la capacidad de generalización de la misma.

De esta forma, MIS-Net se encontraría formada por una estructura de *encoder-decoder* donde la repetición de determinados bloques será clave a lo largo de la arquitectura. Así, en lo que respecta

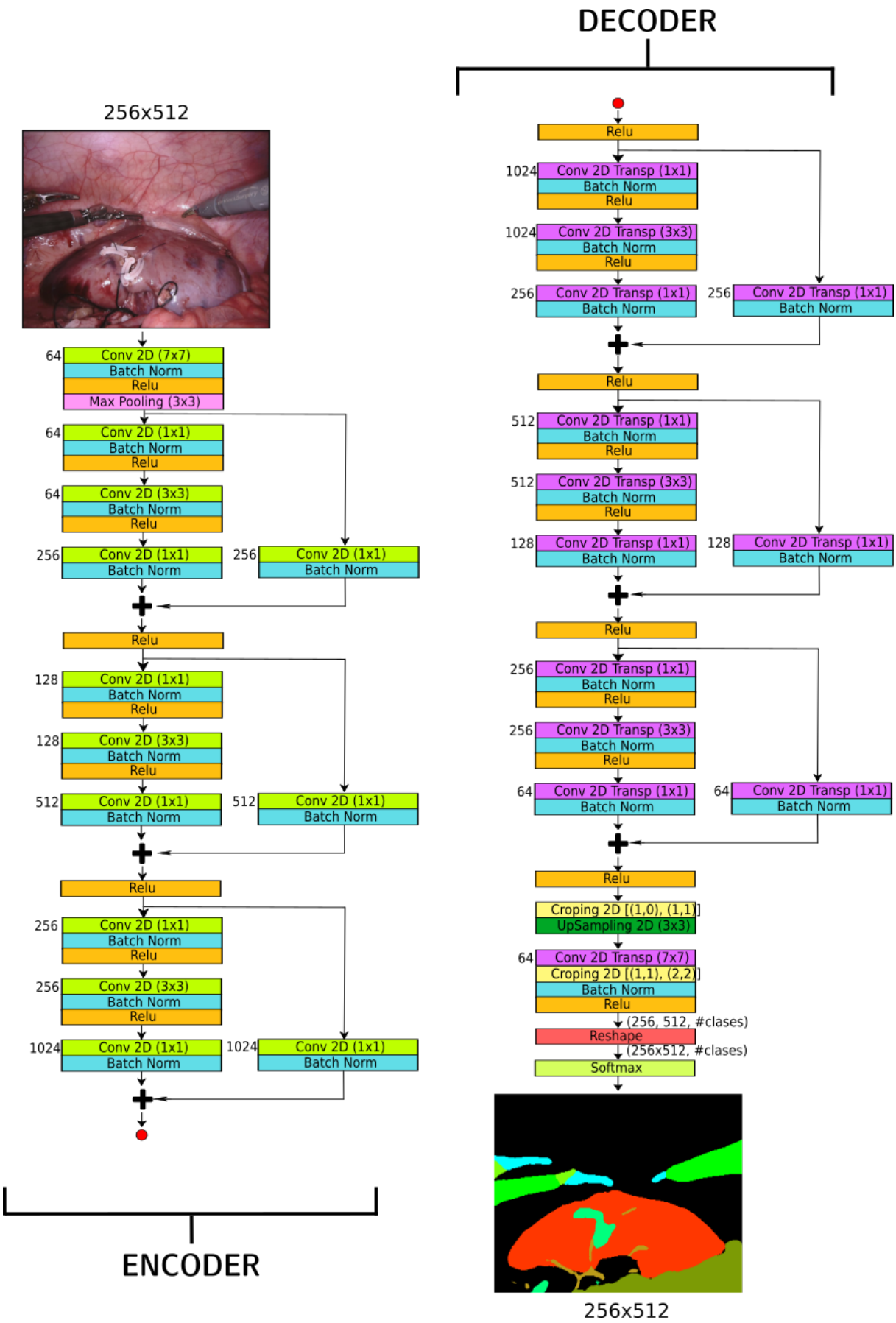


Figura 3.1: Esquema de red MIS-Net

al *encoder*, se caracterizaría por el empleo de un bloque constituido por una sucesión de tres capas convolucionales seguidas de su correspondientes capas de *batch normalization* así como activación RELU. Dentro de esta estructura repetida a lo largo del *encoder*, el único parámetro que se verá modificado será el número de convolucionales realizadas así como el tamaño de los filtros que éstas emplean. De esta forma, según se puede apreciar en la figura 3.1, el número de convolucionales será de 64, 128 y 256 junto unos filtros de dimensiones $1x1$ y $3x3$ indistintamente.

De forma paralela a la incorporación de estos bloques, se añadirá además una estructura residual. El fin que se persigue con esta incorporación es poder disponer de una red con mayor profundidad así como evitar la pérdida de información. En cuanto a la construcción de estos bloques residuales, constarán de una capa convolucional donde el número de operaciones realizadas será de 256, 512 y 1024 disponiendo de filtros de dimensiones $1x1$.

En lo que respecta al decoder, éste seguiría una estructura similar al encontrarse formado por una estructura repetida varias veces junto con un bloque residual en paralelo. De esta forma, la estructura común estaría formada por capas de deconvoluciones donde se llevaría a cabo el proceso inverso a la convolución realizada en el *encoder*. Seguidas a estas deconvoluciones, se encontrarían las respectivas capas de normalización así como de activación RELU. Al igual que en el *encoder*, en este caso el único parámetro que se verá modificado será el número de deconvoluciones realizadas así como el tamaño de los filtros empleados en ellas. Siendo en este caso un total de 1024, 512 y 256 las deconvoluciones realizadas en cada caso con unos filtros de dimensiones $1x1$ y $3x3$ indistintamente.

En cuanto al bloque residual sumado a la salida de la estructura común ya comentada, éste se encontrará formado por una capa de deconvoluciones seguida de su correspondiente normalización. En este caso el número de operaciones realizadas irá aumentado prograsivamente en potencia de 2, donde los filtros empleados serán de dimensiones $1x1$.

Tras obtener la salida proporcionada por el decoder, se llevarán a cabo una serie de operaciones al fin de conseguir unos datos de salida de dimensiones esperadas, *alto x ancho, n° clases*. Por último, con la presencia de una capa Softmax, se proporcionará a la salida las probabilidades de cada una de las clases.

En cuanto a los motivos que impulsaron la creación de una red con una estructura como la mostrada, entre ellos se encuentra el de disponer de una arquitectura que permita una ejecución más rápida dando la posibilidad de que se pueda ejecutar en tiempo real. Para ello, se ha hecho uso de múltiples capas concolucionales que permitan acelerar este proceso. Por su parte, el empleo de una estructura basada en *encoder-decoder* ha sido impulsado en base a que como bien se ha demostrado en [9], es un tipo de esquema que proporciona buenos resultados dentro del campo de la segmentación semántica. En lo que respecta a la incorporación de bloques residuales, el objetivo era poder aplicar una mayor profundidad a la red, ayudando a que no se produjera ninguna pérdida de información.

De esta forma, la combinación de estos aspectos ha derivado en una estructura de red como la presentada en este estudio, donde tanto las entradas como las salidas serían de dimensiones 256 x 512.

3.3 Bases de datos empleadas

Para el desarrollo de este trabajo, se han empleado varias bases de datos de diferente procedencia. De esta forma, dentro del total de bases de datos disponibles, tres en concreto, se organizarán en públicas y no públicas, considerándose dentro de la categoría de públicas, todos aquellos *challenges* que se pueden encontrar en Internet y que están a disposición de la comunidad investigadora.

De esta forma, a lo largo de esta sección se mostrarán cada una de las bases de datos empleadas, especificando de ellas sus características en cuanto a tema, total de imágenes disponibles y reparto de cada una de ellas en función de su empleo para *train*, validación y *test*.

Por último, se detallará el proceso seguido para llevar a cabo un aumento de datos debido al número tan reducido de imágenes de las que se dispone en algunas de las bases de datos empleadas y que por lo tanto puede desembocar en un sobrentrenamiento de la red. Este hecho implica que la red utilizada se aprendería la segmentación empleada en las imágenes de *train* y por lo tanto, para las de *test*, al no haber gran variedad entre ellas, los resultados obtenidos serían bastante buenos, empeorando en este caso en todos aquellos supuestos en los que la diferencia con las imágenes de *train* sea bastante grande. De esta forma, este aumento de datos será empleado en todas las bases de datos utilizadas a fin de garantizar una mayor variabilidad en las imágenes utilizadas.

3.3.1 *Challenges* públicos de imágenes médicas

Dentro de la categoría de *challenges* públicos relacionados con el ámbito médico, en este estudio se han empleado dos de los muchos disponibles. Ambos han sido obtenidos a través de [12], página web en la cual se pueden encontrar tres *subchallenges* diferentes vinculados con las operaciones de cirugía mínimamente invasiva.

De esta forma, los *challenges* disponibles en este caso se encontraría distribuidos en tres subcategorias diferentes correspondientes con el año de publicación de cada uno de ellos. En este caso, para el estudio que compete a este documento, únicamente se ha hecho uso de los *challenges* correspondientes a los años 2015 y 2018, los cuales serán detallados a continuación.

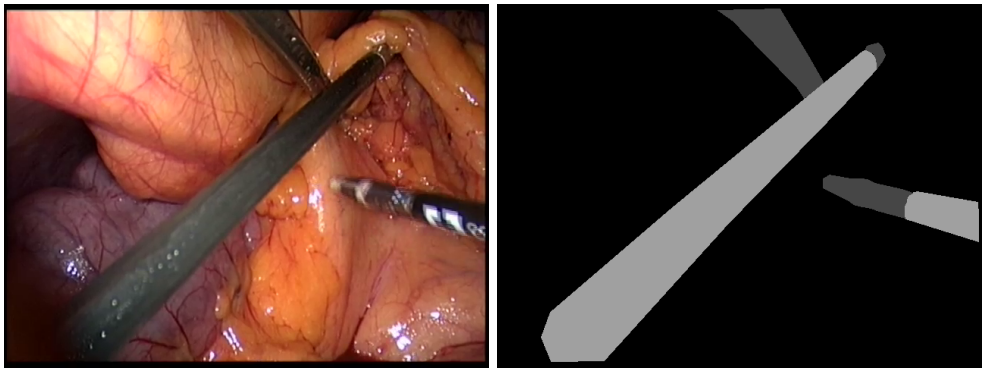
3.3.1.1 Detección de instrumental

Challenge publicado en el 2015 con la finalidad de llevar a cabo la segmentación de instrumentos en intervenciones asistidas por computador debido principalmente al avance que se está produciendo en la visión quirúrgica. Campo que sin duda se está viendo beneficiario de la incorporación de diversas técnicas en las que se lleva a cabo la detección y el rastreo de los instrumentos empleados en las imágenes de laparoscopia. De esta forma, en este caso se hará uso de todas las imágenes pertenecientes al *challenge* en las que se pueda identificar el instrumento rígido que interviene en la operación así como las partes que lo forman, obteniendo en este caso un total de dos clases disponibles, eje y mango del instrumento. A continuación se puede ver un ejemplo de dichas imágenes empleadas en la toma de resultados.

De esta forma, se cuenta, para el proceso de entrenamiento con un total de 4 operaciones diferentes de las cuales se dispone de un conjunto de 40 imágenes de cada una de ellas, disponiendo así de 160 imágenes de entrenamiento semejantes a las mostradas en las figuras 3.2a y 3.2c, las cuales van acompañadas de sus correspondientes Ground Truth (GT), con un aspecto similiar al mostrado en las figuras 3.2b y 3.2d.

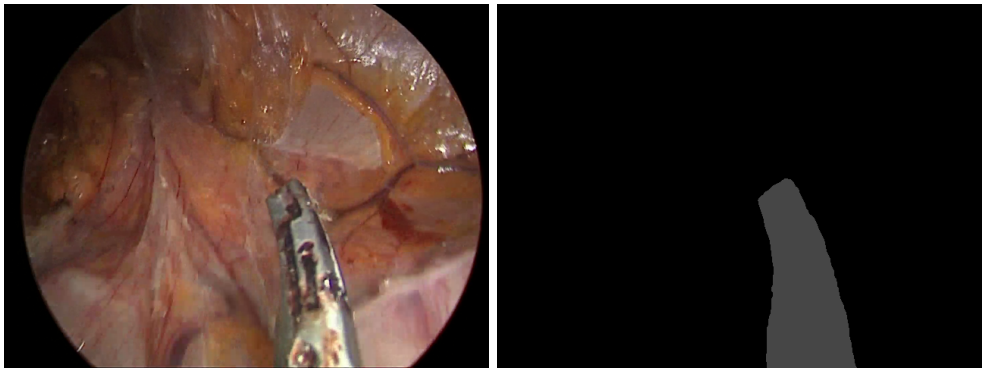
En lo que respecta al *set* de imágenes empleado para el *testeo* de la red, en este caso se dispone de 10 imágenes de las cuatro operaciones utilizadas en el *set* de entrenamiento además de incorporarse dos intervenciones nuevas, las cuales contienen cada una de ellas, 50 imágenes, haciendo así que los datos disponibles para *test* sean un total de 140. Estas imágenes son de un aspecto muy similar a las mostradas en las figuras 3.2.

Por su parte, dentro de las imágenes disponibles para la fase de entrenamiento, un 10 (%) de éstas serán separadas durante el entrenamiento y de forma aleatoria para la fase de validación que se realizará al concluir cada una de las épocas establecidas. Todas las imágenes pertenecientes tanto al *set* de *test* como al de entrenamiento, poseen unas dimensiones de 1280 píxeles de ancho por 1024 de alto.



(a) Ejemplo imagen en RGB para la detección del instrumental

(b) Imagen GT con las clases de la imagen 3.2a



(c) Ejemplo imagen en RGB para la detección del instrumental

(d) Imagen GT con las clases de la imagen 3.2c

Figura 3.2: Ejemplos *challenge* instrumental [12]

Según como se especifica en [28], del etiquetado empleado en este *challenge*, es importante mencionar que es un proceso proporcionado por los autores del propio *challenge* del cual no se ha modificado nada. Estableciendo así la clase 0 para el fondo, empleando para ello una máscara de tres canales, RGB, donde la clase fondo estaría representada con $(0, 0, 0)$. Existirían además dos clases más pertenecientes al propio instrumento, donde se diferenciarían dos partes diferentes, eje y mango, correspondiendo la clase 1 con el mango y siendo representada con la máscara $(70, 70, 70)$ y el eje con la clase 2 y máscara $(160, 160, 160)$.

De forma adicional, el *challenge* da la posibilidad segmentar la imagen en función del tipo de instrumento que intervenga en la operación en lugar de las diferentes partes que lo conforman. En este caso, las clases utilizadas serían cuatro, perteneciendo la 0 al fondo y estando representada con una máscara en RGB de $(0, 0, 0)$, mientras que las clases de la 1 a la 3 pertenecerían a los instrumentos de tipo 1, 2 y 3 que pueden aparecer en la escena, correspondiendo cada sus máscaras con los valores $(20, 20, 20)$, $(40, 40, 40)$, $(60, 60, 60)$ respectivamente. En lo que respecta al tipo de los instrumentos, éste no ha sido especificado con claridad por lo que solo se conocen como tipo 1, 2 y 3.

Aunque esta segunda parte del *challenge* incorpora un total de cuatro clases diferentes a segmentar, se ha optado por emplear la primera de las opciones, en la que el objetivo es identificar las partes del propio instrumento a fin de ver cómo se comporta la red en determinadas zonas de la imagen donde la separación entre clases tiene una mayor importancia a fin de ver el comportamiento en los límites de regiones pequeñas.

Por último, comentar como ya se había indicado, que las imágenes disponibles de este *challenge* sufrirán un proceso de aumento de datos en el que se rotarán y/o trasladarán en función de las opciones. Este aumento de datos solo se realizará en las imágenes que forman el *set* de entrenamiento y por consiguiente, en las de validación, al separarse de éste el 10(%). Aun así, este proceso será detallado en profundidad más adelante.

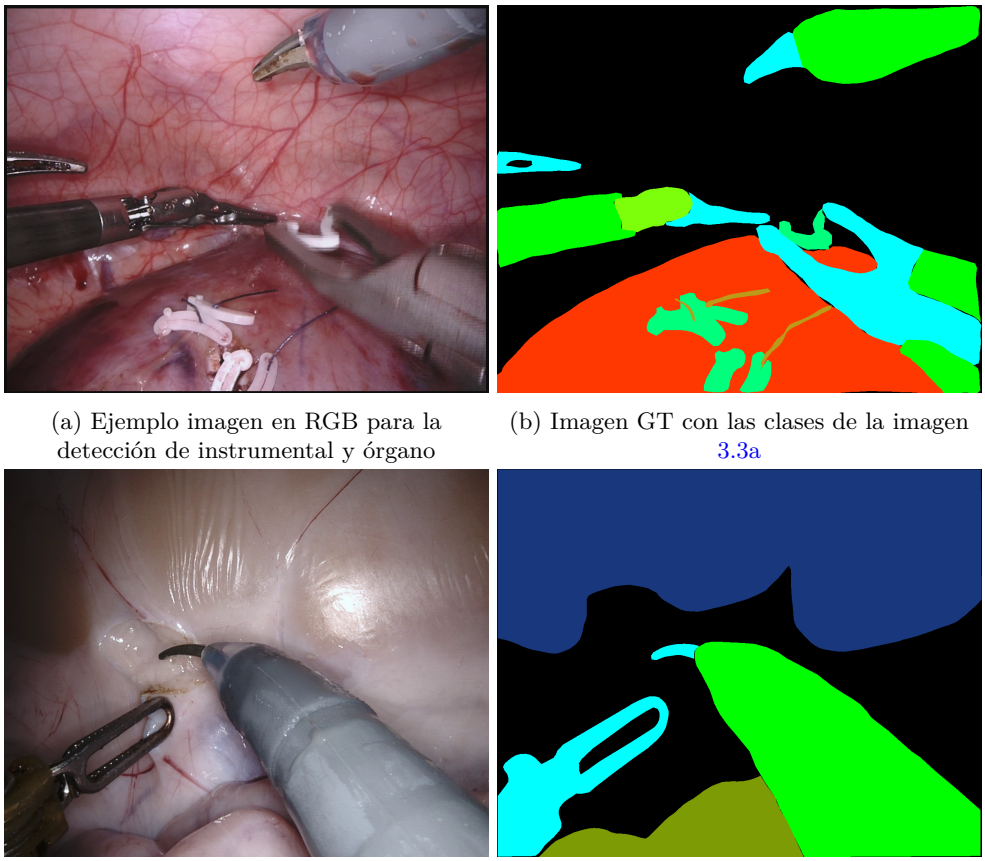
3.3.1.2 Detección de órganos e instrumental en una operación de laparoscopia

Este *challenge* ha sido publicado recientemente a lo largo del verano de este mismo año. Consta de un conjunto de imágenes procedente de 15 procedimientos desarrollado en laboratorios porcinos en los que se han llevado a cabo intervenciones de nefrectomía, en las cuales se ha podido extirpar de forma total o parcial el riñón. Estas intervenciones han sido realizadas todas ellas con el famoso sistemas robótico da Vinci Xi.

Según se especifica en [29], tras la grabación a 60 Hz de cada una de las extirpaciones, se redujo el coste de etiquetado al muestrear dichas grabaciones a una frecuencia de 2 Hz, donde los fotogramas con poco o ningún movimiento fueron eliminadas con el objetivo de formar un conjunto de datos con un total de 149 fotogramas por intervención. De esta forma se dispondrá de un total de 2235 *frames* proporcionados a partir de las 15 necfrectomías realizadas, cuya apariencia será muy similar a las imágenes mostradas en la figura 3.3 donde se puede ver tanto la imagen original como su correspondiente GT, ambas en RGB.

En lo que respecta al etiquetado, éste se llevó a cabo por personal especializado en la anatomía porcina, de tal forma que cada intervención fue etiquetada al completo a mano por una misma persona, estableciendo así un total de 12 clases, numeradas de la 0 a la 11. En cuanto GT correspondiente de cada uno de los *frames* en los que se muestran las diferentes clases que aparecen, éste es incorporado junto a la imagen original, cuya máscara estará expresada también en los tres canales RGB. En la tabla 3.1 se puede ver la lista de clases que han empleado los autores para el etiquetado así como su identificador y los valores de la máscara en formato RGB:

Por su parte, el *set* de *test* constará de un total de 1000 fotogramas distribuidos en cuatro intervenciones diferentes, formadas cada una de ellas por 250 *frames*. Su contenido será también procedente de



(a) Ejemplo imagen en RGB para la detección de instrumental y órgano

(b) Imagen GT con las clases de la imagen 3.3a

(c) Ejemplo imagen en RGB para la detección de instrumental y órgano

(d) Imagen GT con las clases de la imagen 3.3c

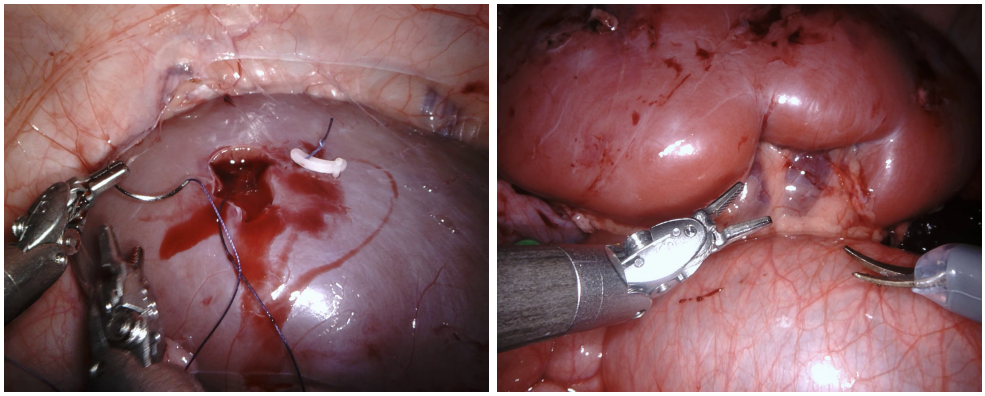
Figura 3.3: Ejemplos *challenge* instrumental y órgano, datos de entrenamiento [12]

Nombre de la clase	Identificador	Máscara en RGB
Fondo	0	(0,0,0)
Instrumento quirúrgico del robot da Vinci: eje	1	(0,255,0)
Instrumento clasper	2	(0,255,255)
Instrumento quirúrgico del robot da Vinci: muñeca	3	(125,255,12)
Parénquima renal	4	(255,55,0)
Fascia renali	5	(24,55,125)
Hilo de sutura	6	(187,155,25)
Abrazaderas	7	(0,255,125)
Agujas de sutura	8	(255,255,125)
Instrumento de succión	9	(123,15,175)
ntestino delgado	10	(124,155,5)
Sonda de ultrasonido	11	(12,255,141)

Tabla 3.1: Clases base de datos instrumental y órgano.

un proceso de necfrectomía pero como una apariencia ligeramente diferente a los *frames* de entrenamiento. En este caso se proporcionan las imágenes obtenidas tanto del ojo derecho como del izquierdo de la cámara. Empleándose para las pruebas los *frames* procedentes del ojo izquierdo.

A continuación, en la figura 3.4 se muestra un par de ejemplo de los datos de *test* de los que se dispone pertenecientes cada una de ellas a los cuatro conjuntos de secuencias proporcionados:



(a) Ejemplo imagen RGB, operación 1 (b) Ejemplo imagen RGB operación 3

Figura 3.4: Ejemplos *challenge* instrumental y órgano, datos de *test* [12]

En cuanto a las dimensiones de cada uno de los fotogramas empleados, todos son de 1280 x 1024 píxeles y en todos los casos se proporcionan tanto las imágenes obtenidas con el ojo izquierdo de la cámara así como con el derecho, siendo únicamente la perteneciente al lado izquierdo la que corresponderá con las etiquetas facilitadas por los autores.

Por último, comentar que este *challenge*, además de utilizarse para ver los resultados obtenidos con la red propuesta y compararlos con los proporcionados por otras arquitecturas conocidas, será empleado también para hacer *fine-tuning* y entrenar de esta forma, la bases de datos propia a partir de los pesos obtenidos para este *challenge*. En cuanto al concepto de *fine-tuning*, se profundizará más en detalle en el apartado correspondiente, sección 3.4.5

3.3.2 *Challenge* propio

En lo que respecta a esta base de datos, tiene la característica de no estar disponible para todo el mundo pues se trata de una recopilación de las grabaciones correspondientes a 10 pacientes. Dichas intervenciones fueron llevadas a cabo en el hospital *Chu Estaing* de Clermont-Ferrand en colaboración con el grupo *EnCoV* de la Universidad Clermont Auvergne en Francia.

En este caso, el órgano intervenido fue el hígado en el cual se examinaba su estado, pudiendo además, en el caso de presencia de tumor, llevar a cabo la extirpación de éste. Estas intervenciones se realizaron a través de una operación de laparoscopia, la cual, como bien se ha detallado en la introducción de este documento, representaría una de las técnicas que hoy en día se consideran parte de la cirugía mínimamente invasiva. Recordar que este tipo de cirugía se caracterizaba por realizar un pequeña incisión en el paciente de entre 0.5 a 1 cm, de forma que permita poder introducir a través de ese agujero un pequeño instrumento tubular con el que comunicarse con el interior de dicho paciente, para, posteriormente, a través de dicho tubo, introducir el instrumental necesario para llevar a cabo la operación así como una pequeña cámara con la que poder monitorizar todo el proceso.

De esta forma, a lo largo de la recopilación de grabaciones de las diferentes operaciones disponibles, se pudieron encontrar grabaciones de distinta duración así como tamaño, en las cuales se podía apreciar

tanto el proceso de extirpación del tumor, como de cauterización de una determinada zona o de simple inspección tanto del propio hígado como del resto de partes que forman el hígado o lo rodean, como pueden ser la vesícula biliar o el ligamento que une dicho hígado con la pared del propio cuerpo del paciente. De esta forma, se seleccionaron los *frames* más representativos de cada una de las grabaciones correspondientes a cada paciente con intención de evitar la similitud entre varios *frames* consecutivos. Este hecho ha sido un poco difícil pues en muchos casos entre un número determinado de *frames* las variaciones producidas eran mínimas, lo que dificultaba el hecho de tener un *set* de datos rico en variedad. Aun así se ha hecho una selección exhaustiva de los *frames* que componían cada una de las grabaciones de los diferentes pacientes, eliminando todos aquellos en los que había demasiado movimiento de la cámara y apenas se podía identificar ninguna zona concreta. Por su parte, los *frames* en los que no se veía hígado sino proceso de preparación o inserción de la cámara en la zona tubular, también han sido eliminados del proceso de selección junto con los que mostraban la fase de la intervención en la que el hígado empezaba a ser cortado perdiendo así la forma básica que suele tener o aquellos en los que se mostraba el proceso de extirpación del tumor, al perder el hígado su forma habitual y al llenarse la escena de demasiada sangre y/o agua empleada para limpiar la zona.

Es por eso que al hecho de disponer de un número muy limitado de pacientes y que por lo tanto la variedad de los hígados disponibles para entrenar a la red sea muy reducida, se junta el hecho de que no todos los *frames* han sido aprovechables bien por no tener un contenido apropiado para el objetivo perseguido en este estudio o bien por evitar la similitud entre demasiados *frames* consecutivos que pudiera desembocar, de alguna forma, al aprendizaje por parte de la red de los datos de entrada, haciendo que no sea capaz de diferenciar fuera de unos datos semejantes a esos *frames* y por lo tanto realizando una mala segmentación en ellos.

Con esto, el total de *frames* que forman esta base de datos sería de 614, un número demasiado pequeño para obtener unos resultados buenos sin que la red llegue al punto de aprenderse los datos de entrenamiento. Aun así, se ha seguido con el proceso a fin de ver los resultados obtenidos y poder llegar a unas conclusiones a cerca de esta base de datos.

De esta forma, una vez se dispone del total de *frames* que formarían el conjunto de datos correspondiente a operaciones de laparoscopia en hígados humano, se establecieron unas dimensiones comunes a todos los *frames*, con el fin de evitar la variabilidad de tamaños en el conjunto de datos. De esta forma, las dimensiones seleccionadas fueron, 1280 x 1024 píxeles, tamaño más repetido en la mayoría de grabaciones disponibles.

Así, tras fijar las dimensiones, se llevó a cabo el proceso de etiquetado manual de cada uno de los *frames*. Para ello, se optó por determinar las clases que entrarían en juego en la segmentación de los diferentes *frames*. Para ello, analizando todos los fotogramas disponibles y la anatomía propia de los seres humanos, se optó por definir 7 clases diferentes, las cuales serían:

Nombre de la clase	Identificador
Fondo	0
Hígado	1
Instrumental	2
Ligamento	3
Grasa	4
Vesícula biliar	5
Sangre	6

Tabla 3.2: Clases base de datos propia.

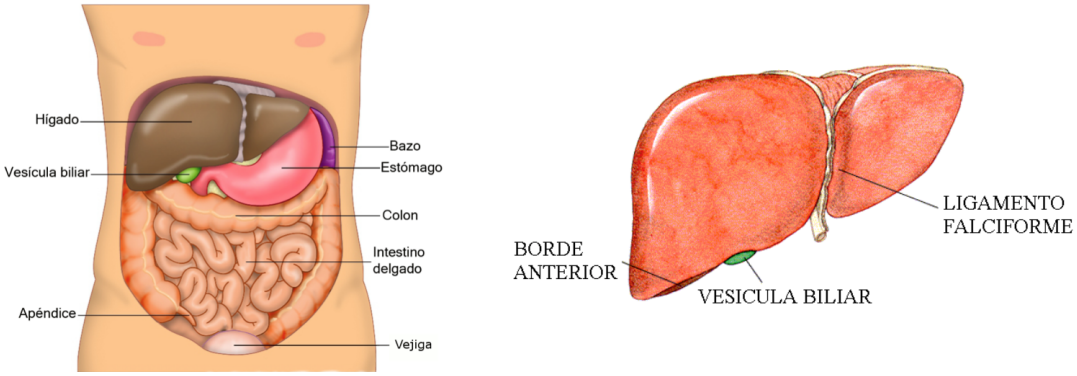


Figura 3.5: Anatomía del cuerpo humano, hígado

Para facilitar la imagen visual sobre la disposición de los diferentes órganos que componen el interior de los humanos, se puede observar la siguientes figura 3.5, donde se aprecia la distribución, no solo del hígado, sino también de la vesícula o el ligamento a lo largo del interior del cuerpo. Conociendo así la posición exacta de cada uno de ellos.

Así, tras la definición de las diferentes clases que conformarían la segmentación de los distintos *frames*, se llevó a cabo el proceso de etiquetado, para el cual se empleo la toolbox de *Matlab*®, ***Image Labeler***, de la cual se puede encontrar toda la información pertinente en la correspondiente documentación proporcionada por *Matlab*®, [30].

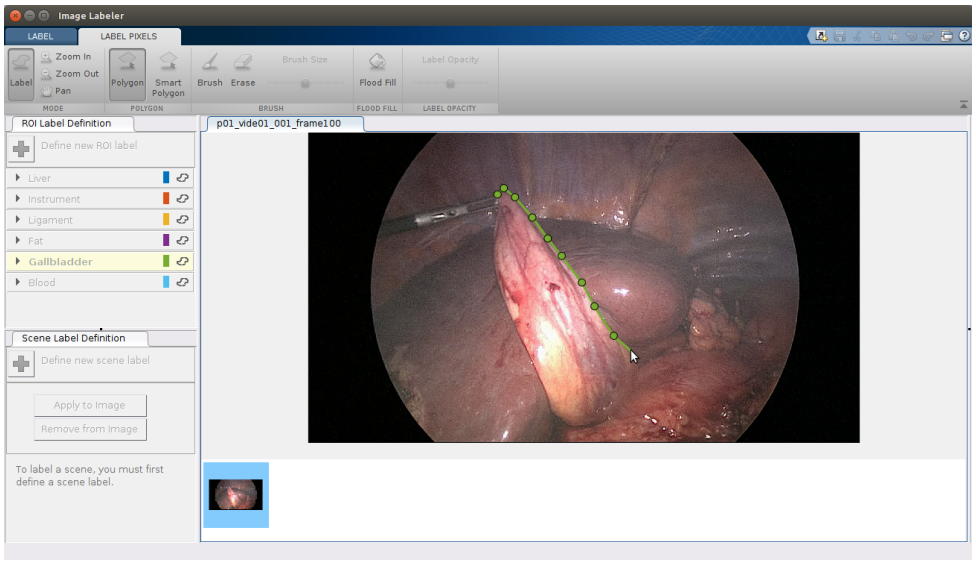
Esta *Toolbox* da la posibilidad de definir una serie de etiquetas a las cuales se les puede asignar un color específico y crear así un archivo *.mat* que podrá ser utilizado cada vez que se quiera etiquetar un nuevo frame. Así, tras la asignación de un color específico a las diferentes clases definidas, se procedió a iniciar la fase de etiquetado, proceso fácil en cuanto a ejecución pero costoso en cuanto a tiempo invertido así como por el cuidado extremo que requiere un correcto etiquetado, en el que es necesario que todas las clases se encuentren correctamente definidas. Este hecho lleva consigo implícito que además de inspeccionar las imágenes con un cierto grado de especialidad, hubo que documentarse de forma básica sobre la anatomía humana a fin de evitar posibles confusiones en cuanto a etiquetas erróneas. De esta forma, haciendo uso de la aplicación citada, se inició el proceso de una forma muy similar a la mostrada a continuación en la siguiente secuencia de imágenes:

En lo que respecta a las figuras 3.6a y 3.6b se puede ver cómo se llevó a cabo la segmentación de la clase 5 correspondiente con la vesícula biliar y cómo, siguiendo el mismo procedimiento, se llega a obtener un etiquetado completo del frame objeto de segmentación como el mostrado en la figura 3.6c.

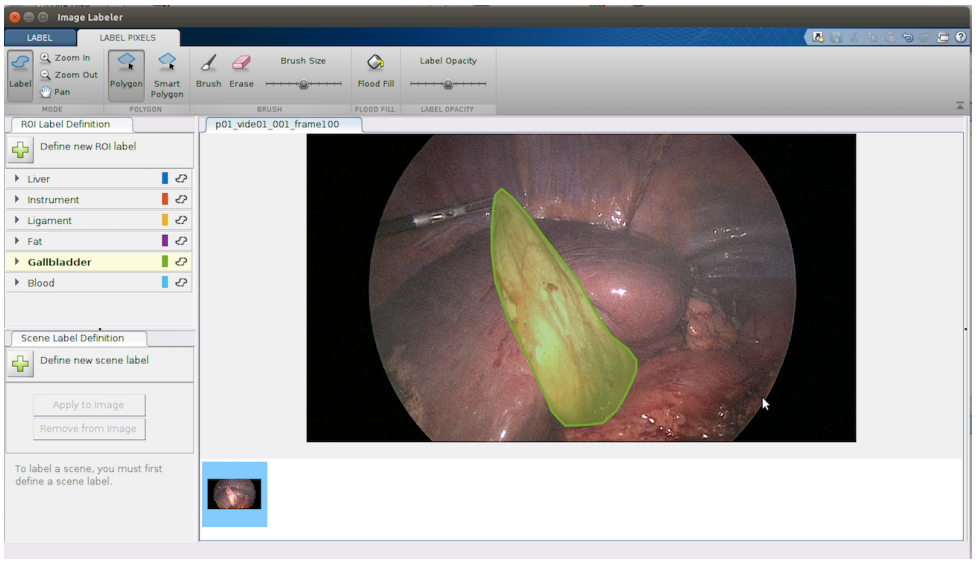
Una vez concluido el etiquetado, al salvar los resultado, la *toolbox* proporciona un archivo *.mat* nombrado como *gTruth.mat* en el que se especifica el GT correspondiente a la imagen inicial. Además, proporciona una imagen *png* en escala de grises con tres canales de color en los cuales, el valor de color será el mismo para todos, apareciendo de esta forma, cada una de las clases etiquetada mediante la *toolbox*. A continuación, en la figura 3.7 se puede ver el resultado obtenido tras el proceso de etiquetado.

Una vez obtenida el etiquetado completo de cada uno de los *frames* que formarían el conjunto total de datos, estas imágenes se utilizarán como GT de los *frames* originales para proporcionar ambos a la red y comenzar así el entrenamiento.

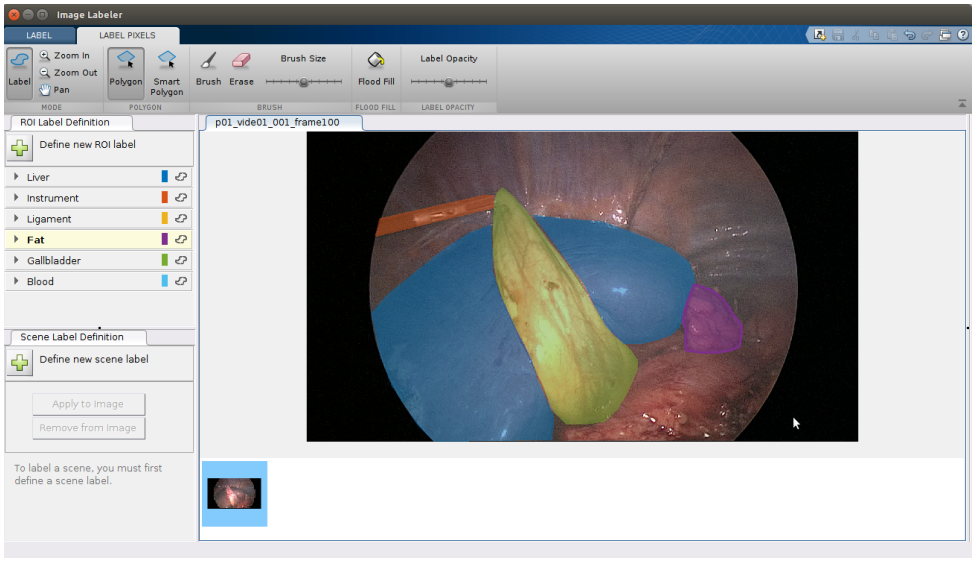
En cuanto al reparto realizado para el conjunto de *train*, validación y *test*, éste se ha hecho de forma manual, separando aproximadamente del conjunto total, un 10 (%) para la fase de validación, así como 16 imágenes para el conjunto de *test*, quedando de esta forma, un *set* de entrenamiento formado por 539 *frames*, siendo ésta una cantidad demasiado pequeña.



(a) Inicio del proceso de etiquetado



(b) Etiquetado de la clase 5 correspondiente a Vesícula Biliar



(c) Fin del etiquetado del frame completo

Figura 3.6: Proceso de etiquetado

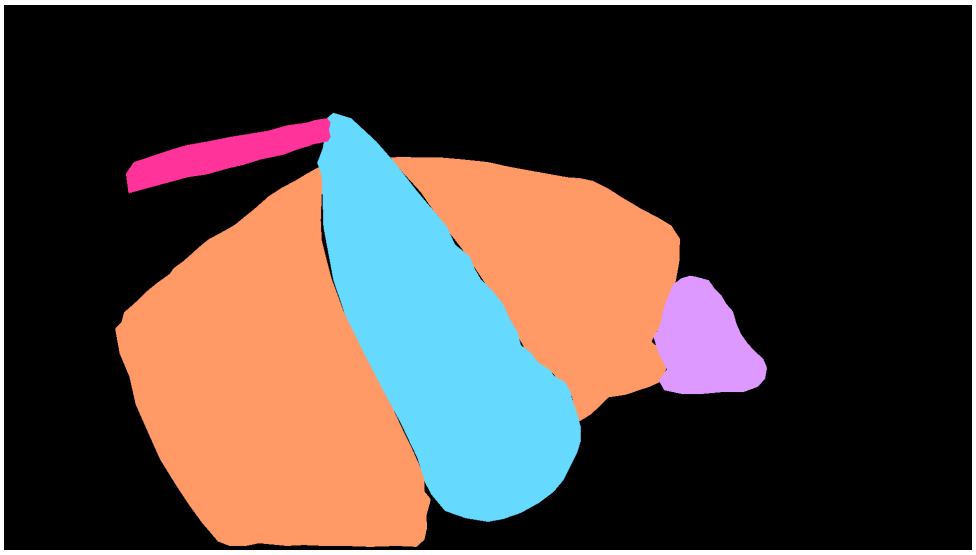


Figura 3.7: Imagen etiquetada mediante la *toolbox ImageLabeler*

En lo que respecta a la separación del conjunto total en tres subconjuntos, es importante mencionar que a la hora de crear el *set* de validación, éste ha sido creado partiendo de la idea de evitar que varios *frames* consecutivos se encuentren en la fase de validación a fin de que entre ellos sean lo más diferentes posibles con el objetivo de evitar ese aprendizaje de la red.

Con respecto a los datos de *test*, estos han sido elegidos de forma aleatoria entre el resto de *frames* que quedaban, creando así una lista de números *random* y seleccionando los *frames* que correspondieran con los valores de dicha lista con el fin de garantizar la aleatoriedad en el *testeo* de los resultados obtenidos.

De esta forma, el aspecto de las imágenes procedentes de operaciones de hígado mediante laparoscopia en 10 pacientes diferentes, tendrían un aspecto similar al mostrado a continuación en las figuras 3.8, donde se puede apreciar en cada caso, no solo la imagen en RGB sino también su correspondiente GT en escala de grises.

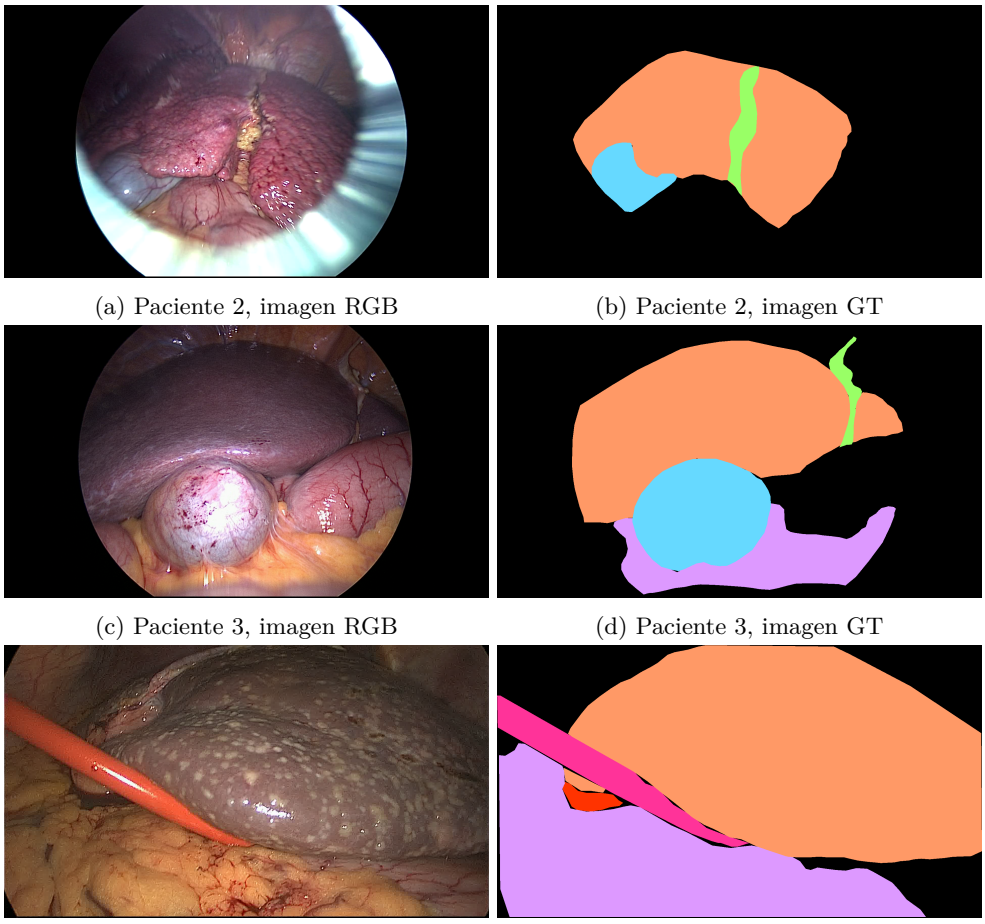
Como se puede observar en la figura 3.8a con su correspondiente GT representado en la figura 3.8b, a la hora de seleccionar los *frames* también se ha tenido en consideración aquellos en los que se aprecie parte del tubo empleado para la inserción tanto de los instrumentos como de la cámara a fin de que la red vea también imágenes de ese tipo.

Debido al problema de la cantidad de *frames* y de la poca variedad entre ellos, en esta base de datos es de especial importancia la presencia del proceso de aumento de datos ya que con él, se intentará obtener puntos de vista diferentes a los habituales en los *frames* originales. Este aumento de datos se empleará solo y exclusivamente en los *frames* que formen parte del conjunto de entrenamiento

3.4 Estrategias en el entrenamiento de la red

Tras la preparación de las diferentes bases de datos ajustando el etiquetado y las dimensiones de las imágenes acorde a las entradas de la red, se comienza con la fase de entrenamiento. Esta parte es común a todos las *challenges*, independientemente de su procedencia, pues solos algunos parámetros variarán dependiendo del tipo de imagen que se pretenda segmentar.

Durante el entrenamiento, se probará tanto la propuesta de red presentada en este capítulo, MIS-Net, así como las arquitecturas más significativas dentro del campo de la segmentación semántica como pueden ser la UNet o la SegNet, ya detalladas en la sección 2.4.2.



(a) Paciente 2, imagen RGB (b) Paciente 2, imagen GT

(c) Paciente 3, imagen RGB (d) Paciente 3, imagen GT

(e) Paciente 4, imagen RGB (f) Paciente 4, imagen GT

Figura 3.8: Ejemplos de algunos *frames* de la base de datos

Dentro de la fase de entrenamiento, hay un concepto importante que define a una red como es la *función de costes o función de pérdidas*. Este término se puede definir como la penalización que se otorga a todas aquellas soluciones peores que la solución óptima, la cuál se caracteriza por tener coste cero. De esta forma, para saber cuál es la pérdida obtenida por cada una de las soluciones proporcionadas, habrá que emplear una función de costes que se adecue al problema que se está tratando. Es por eso, que en este caso se ha optado por emplear **Cross-entropy** como función de pérdidas.

Cross-entropy se caracteriza por medir el rendimiento de un modelo de clasificación cuyo resultado es un valor de probabilidad entre 0 y 1 [31]. En cuanto al valor, éste aumentará a medida que la probabilidad obtenida diverja del resultado real de cada clase y disminuirá a medida que se vaya acercando al valor real, siendo en este caso un modelo perfecto el que obtuviera una función de pérdidas de 0.

Así, la función de pérdidas *Cross-Entropy* se puede expresar mediante la siguiente ecuación 3.1: [32]

$$\mathcal{L}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(p_i) + (1-y_i) log(1-p_i)] = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} log(p_{ij}) \tag{3.1}$$

Donde en este caso, i representaría las muestras de cada observación y j cada una de las clases. Por su parte y representaría la etiqueta correspondiente a una determinada región. Por último, indicar que las diferentes probabilidades p_{ij} estarán comprendidas en el conjunto de valores entre 0 y 1, siendo de esta forma $p_{ij} \in (0, 1)$.

En lo que respecta a la fase previa al entrenamiento, hay que mencionar que se llevará a cabo una serie de procesos preparativos entre los que se pueden encontrar el balanceo y ajuste de clases así como el aumento de datos, los cuales serán detallados a continuación:

3.4.1 Ajuste de clases

Debido a la diversidad de etiquetas entre las diferentes bases de datos a utilizar, es necesario que todas ellas sean unificadas de alguna forma a fin de que el entrenamiento se pueda realizar de forma correcta. De esta forma, como bien se ha visto en la sección 3.3, no todas las bases de datos tienen las mismas etiquetas al haber sido realizadas por personas diferentes. Tanto es así, que en lo que respecta al *challenge* público para la detección de instrumentos así como órganos en intervenciones quirúrgicas en cerdos, las etiquetas están establecidas según los valores en RGB que tendrían los colores que representan las diferentes clases.

Por su parte, en lo que respecta al otro *challenge* público del que se hace uso, empleado para la detección del instrumental que interviene en una operación de cirugía mínimamente invasiva, las etiquetas que corresponden con cada clase están, en su origen, representadas por los tres canales RGB pero en en escala de grises. Así, se tendrían las clases 0, 70 y 160, correspondientes con los 3 tonos de grises empleados.

En cuanto a la base de datos propia etiquetada de forma manual, en este caso cada clase estaría enumerada del 0 al 6 empleando para ello un único canal, método muy diferente al empleado en las otras dos bases de datos. De esta forma, con la intención de unificar el etiquetado en todos los *challenges*, se ha optado por modificar las etiquetas de los *challenges* públicos por ser la opción más fácil.

Así, en lo que respecta a la modificación realizada, se han llevado a cabo los siguientes ajustes dependiendo del *challenge* involucrado:

- **Instrumental y órganos**: En este caso el procedimiento seguido ha consistido en la creación de tantas máscaras como clases disponibles. En cada máscara, se ha especificado el valor de cada uno de los canales RGB correspondiente con la clase a ajustar, de esta forma si se pretende modificar la clase 3, *Instrumento quirúrgico del robot da Vinci: muñeca*, la máscara creada tendrá los siguientes

valores para cada uno de los colores, $(125, 255, 12)$, de tal forma que mediante el uso de dicha máscara se pueden localizar las zonas de la imagen original en las que existe esa clase para modificar posteriormente esos valores por el identificador correspondiente a dicha clase 3.

Este proceso se repetiría con cada una de las clases existentes y al final se obtendría una imagen en la que los valores correspondientes al color que lo representa estarían sustituidos por el correspondiente identificador de cada clase, teniendo así tres canales con el mismo valor.

Por último, como paso previo a la entrada de la red, bastaría con coger un único canal de los tres disponibles y seguir con el entrenamiento de forma normal, pues en lo que respecta a este *challenge*, ya tendría unas etiquetas con una anotación similar a la utilizada en la base de datos propia.

- **Instrumental**: En lo que respecta a este *challenge*, la anotación es ligeramente diferente al resto. En este caso, las imágenes de *Ground Truth* contendrían la regiones que forman cada una de las posibles clases, la cual recordemos que podría ser 3, fondo, mango y eje del instrumento, estando éstas identificadas con los valores 0, 70 y 160 respectivamente, haciendo uso de los tres canales de color.

  Para obtener una anotación similar a las otras dos bases de datos, se ha optado por modificar el identificador de las clases 70 y 160 por 1 y 2 respectivamente. De esta forma, haciendo uso de unas máscaras que permitan la búsqueda de esos valores a lo largo de la imagen se podría llevar a cabo el cambio deseado con un funcionamiento parecido al expuesto en el caso anterior, consiguiendo así una nueva anotación basada en los identificadores 0, 1 y 2, pero manteniéndose en los tres canales de color. Por último, al igual que en el *challenge* anterior, bastaría con seleccionar uno de esos canales para comenzar el entrenamiento.

De esta forma, la fase de *ajuste de clases*, se podrá hacer de forma mecánica con el simple hecho de indicar previamente a la red la base de datos que se va a emplear para el entrenamiento. Este hecho se realizaría con el simple uso de una bandera al inicio del proceso, la cual estaría a *true* para el *challenge* deseado y a *false* para el resto

3.4.2 *Data aumentation*

El reducido número de *frames* que contienen algunas bases de datos como el *challenge* público en el que se pretende segmentar las partes de los instrumentos que formen parte de la intervención quirúrgica así como la base de datos propia en la cual se trata el problema de segmentación a partir de una operación de laparoscopia en hígados humanos es una dificultad a la que hay que hacer frente si se quieren conseguir unos resultados coherentes sin que la red llegue a aprenderse los *frames* de la fase de *train*.

Como solución a dicho problema, se ha optado por llevar a cabo una fase de aumento de datos previa al proceso de entrenamiento de tal forma que se garantice que a partir de los *frames* originales, se van a crear, de forma aleatoria, ciertos *frames* con un determinado ángulo de giro, una determinada traslación posible tanto en el eje x como en el eje y, así como una leve distorsión de la imagen que asemeje las situaciones en las que por cualquier motivo, la lente de la cámara empleada para monitorizar la intervención puede no proporcionar una imagen lo suficientemente claro dando lugar a unos *frames* con una cierta borrosidad.

De esta forma, dentro del aumento de datos se van a producir tres alteraciones en las imágenes originales, rotación, traslación y aplicación del filtro Blur, las cuales serán aplicadas, dependiendo de la alteración, tanto a la imagen RGB así como a su correspondiente GT a fin de intentar proporcionar una mayor variedad a la entrada y evitando en cierto modo, un aprendizaje casi por completo de los datos disponibles al ser estos de una cantidad tan pequeña. Así, el proceso seguido en la aplicación de las variaciones en los diferentes *frames* ha sido el siguiente:

- **Rotación**: Estará comprendida entre $-5°$ y $5°$ de tal forma que se permita el giro de la imagen tanto a derechas como a izquierdas una vez calculado el punto medio de dicha imagen. El valor del ángulo a girar será elegido de forma aleatoria en cada uno de los *frames* de entrada y será siempre con variación de una unidad, produciéndose por lo tanto en todos los casos, ángulos enteros a girar.

- **Traslación**: El proceso de traslación es llevado a cabo de una forma muy similar al de rotación solo que en este caso se elegirá en cada frame de forma aleatoria los píxeles a desplazarse para el eje x así como para el eje y. En ambos casos el rango será de -10 a 10, cubriendo así todos los sentidos, arriba-abajo y derecha-izquierda. En la traslación, así como en la rotación, las variaciones será de una unidad por lo que los desplazamientos se producirán de píxel en píxel en todos las direcciones.

- **Aplicación de una ROI**: Una vez realizada la rotación así como la traslación, antes de aplicar una leve distorsión en la imagen, se seleccionará una ROI de la propia imagen a fin de eliminar los posibles píxeles en negros obtenidos tras los cambios de posición de la imagen, píxeles que al ser representados en RGB con $(0, 0, 0)$ serían considerados como clase fondo, pudiendo perjudicar el baremo de clases realizado durante el entrenamiento con el objetivo de dar pesos a las clases en función de su frecuencia de aparición. Este baremo será comentado en detalle en la parte de entrenamiento. De esta forma, la región a seleccionar se elegiría de dimensiones acorde a cada una de las bases de datos probadas, ya que recordemos que no todas ellas tenían *frames* con el mismo tamaño. Garantizando así que esos pocos píxeles incluidos tras el desplazamiento fueran descartados al disminuir ligeramente el tamaño de la imagen.

- **Filtro Gaussian-Blur**: Es la única de las alteraciones que no se realiza sobre el GT de la imagen de entrada, el resto de variaciones sí. En este caso lo que se busca es desenfocar la imagen a través de un filtro que emplea una función gaussiana con la que se calcula la trasnformación que hay que aplicar a cada píxel que conforma la imagen, provocando así una sensación de borrosidad. De esta forma, la ecuación de una función gaussiana en dos dimensiones quedaría representada según se muestra en 3.2 : [33]

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-} \frac{x^2 + y^2}{2\sigma^2} \tag{3.2}$$

  Una vez conocida la expresión de una función gaussiana en dos dimensiones y haciendo uso de las librerías de OpenCV disponibles, se puede implementar un filtro gaussiano-Blur de forma muy sencilla con solo especificar el tamaño del *kernel*, ya que si no se indica nada más, la propia función interpretará que el cálculo de σ se realizará a partir de las dimensiones del propio *kernel*. La función empleada en este caso es *cv2.GaussianBlur()*, cuya documentación sobre su funcionamiento se puede encontrar en [34].

  En cuanto a las dimensiones del *kernel*, éstas serán elegidas de forma aleatoria entre una sucesión de valores especificada previamente, siendo éstos: 1, 3, 5 y 7. En este caso, un *kernel* de 1 implica que no se produciría sensación de borrosidad en la imagen mientras que a media que aumente el valor, la distorsión será cada vez mayor, siendo por lo tanto, 7, el valor menos apropiado. En base a esto, a la hora de proporcionar uno de los cuatro valores disponibles, se ha optado por asignar a cada uno de ellos una probabilidad de repetición, lo que indicaría que cuanto menor sea esta probabilidad, menos veces saldrá dicho valor como tamaño del *kernel*. De esta forma, las probabilidades empleadas fueron 0.5, 0.3, 0.15 y 0.05 respectivamente, observándose que a medida que se incrementa el valor del *kernel*, la probabilidad de repetición es cada vez menor.

Es importante mencionar que en los tres casos, se ha intentado que las modificaciones producidas en la imagen no sean excesivas y puedan, de esta forma, simular un frame que podría existir perfectamente

tras un ligero movimiento o giro de la cámara encargada de la grabación o simplemente que dicha imagen pudiera estar más borrosa o menos según las condiciones que se estuvieran dando durante la intervención.

Por último, es importante mencionar que este aumento de datos se realiza únicamente en el conjunto de datos de entrenamiento ya que el de validación haría uso de los *frames* tal cual, sin sufrir ninguna variación.

3.4.3 Balanceo de clases

En las redes de segmentación, el balanceo de clases es una fase importante para poder mejorar los resultados. Este proceso consiste principalmente, en realizar un conteo de las clases en cada uno de los *frames* que forman parte del entrenamiento con el objetivo de identificar las clases más frecuentes frente a las menos y asignar así niveles de importancia, empleados por la red para prestar atención a aquellas clases menos frecuentes y que por lo tanto tienen un nivel de importancia mayor.

Esta idea se aplicó además en el entrenamiento de la SegNet [9], donde se estableció unos pesos para cada clase existente. De esta forma, siguiendo esa idea, en este caso en lugar de definir los pesos para cada clase, al tratarse de un entrenamiento compartido por varias bases de datos y para evitar una mala ponderación de las clases, se optó por calcular dicho balanceo de una forma previa al entrenamiento.

De esta forma, el cálculo se realizó haciendo uso de los histogramas de cada frame de entrenamiento de tal forma que cada histograma calculado con la llegada de un nuevo frame, fuera almacenado en un acumulador. A partir de ese acumulador se podría saber el número de píxeles totales que forman el *set* de entrenamiento así como las clases más habituales y las menos frecuentes. Con esta información se podría obtener de esta forma, la ponderación correspondiente para cada una de las clases existentes en cada base de datos con solo aplicar la siguiente expresión:

$$Array_{ponderacion}[n_{clases}] = \frac{Total_{pixeles}}{Clases * acumulador_{histogramas}[n_{clases}]} \tag{3.3}$$

Donde $acumulador_{histogramas}[n_{clases}]$ representaría el *array* con la suma de todos los histogramas realizados de cada uno de *frames* en los que se especificaría el número de píxeles de cada clase. Como resultado, se obtendría otro *array* con la ponderación final para cada clase en función de los píxeles que contenga, siendo mayor cuanto menor sea esta cantidad.

Para facilitar los resultado, las ponderaciones finales han sido multiplicadas por un factor, calculado previamente, que permita asignar a la calse 0, correspondiente con el fondo, una ponderación de 1. Esto se ha hecho así para garantizar que la clase **fondo**, la más numerosa en todas las bases de datos al contener un número de píxeles superiores al resto de clases, tenga siempre una ponderación de 1, siendo éste el máximo valor alcanzable y estando por lo tanto, el resto de ponderaciones por debajo de este valor.

Una vez que se dispone del *array* final con la ponderación de cada clase, solo será necesario indicarle al software que tenga en cuenta estos valores. Como en este caso se hace uso de Keras para desarrollar toda la parte de construcción, entrenamiento, validación y *testeo* de la red, entre sus librerías aptas para *deep learning* se proporciona la opción de emplear dos módulos diferentes para realizar el entrenamiento, *fit* y *fit generator*, los cuales contemplan la posibilidad de indicarles un vector con la ponderación de clases como bien se puede ver en la documentación propia de Keras para este caso [35].

3.4.4 Optimizadores empleados

Dentro del proceso de entrenamiento llevado a cabo para un modelo de red determinado del cual se pretender conseguir los mejores resultados posibles, la elección a la hora de emplear un tipo u otro de optimizadores es de vital importancia.

Por su parte, el proceso de optimización se puede definir como un conjunto de algoritmos que ayudan a minimizar o maximizar una función objetivo o función de pérdidas. Esta función dependerá de aquellos parámetros internos del modelo que son empleados para la obtención de los valores de salida correspondientes a dicha función objetivo, para lo cual, se lleva a cabo un aprendizaje de estos parámetros propios al modelo. En este caso, tanto los pesos empleados (W) así como el *bías* (b) de la red, son parámetros internos que se van aprendiendo y actualizando a medida que el resultado se va acercando a una solución óptima que permita, de este modo, minimizar la función de pérdidas. [36]

De esta forma, debido a la importancia que tiene el correcto aprendizaje y su correspondiente actualización de estos parámetros internos es muy importante llevar a cabo una buena elección en cuanto a la fase de optimización se refiere, pues de ella dependerá parte del resultado obtenido. Tanto es así que a lo largo del campo de la optimiación se pueden encontrar multitud de algoritmos que se encargan de intentar conseguir la minimización de la función de pérdidas de la mejor forma posible. En cuanto a información se refiere de los posibles métodos de optimización se puede encontrar documentación al respecto en [36] así como [37] entre otras muchas.

Con esto, los métodos de optimización se pueden clasificar acorde a dos categorías: Algoritmos de optimización de primer orden y algoritmos de optimización de segundo orden. En cuanto a los primeros, estos se caracterizan por el empleo de la derivada de primer orden, la cual proporcionaría información acerca de si la función está aumentando o disminuyendo en un determinado punto, sabiendo así si la función de cortes está acercándose a su mínimo o no. Dentro de esta categoría, sin duda, el algoritmo de optimización más empleado es el *Descenso por Gradiente*. Por su parte, en cuanto a los optimizadores de segundo orden se refiere, éstos se caracterizan por el empleo de la derivada de segundo orden para saber si la función de pérdidas está siendo minimizada o maximizada. Esta derivada también es conocida como *hessiano* y estaría representada por una matriz de derivadas parciales de segundo orden. Debido a la dificultad que conlleva el cálculo de la derivada de segundo orden, ésta normalmente no se suele emplear mucho. [36]

De esta forma, tras conocer el principio en el que se basan los algoritmos de optimización así como las dos posibles categorías en las que se pueden dividir, se procede a entrar en detalle sobre alguno de los optimizadores más empleados y los cuales son de gran importancia en este estudio, pues han sido con los que se han realizado las fases de optimización en la búsqueda del mínimo valor posible para la función de pérdidas. Con esto, los dos algoritmos de optimización que se detallarán a continuación serán Stochastic Gradient Descent (SGD) así como Adam.

SGD: Partiendo de los conocimientos desarrollados en [36], SGD es una variante del conocido Gradient Descent (GD) donde se realiza el cálculo del gradiente en todo el conjunto de datos disponible hasta encontrar el valor mínimo, para luego, actualizar los parámetros internos al modelo. En este caso, al realizarse el cálculo sobre todo el conjunto de datos, puede provocar que si éste es demasiado grande, el proceso pueda resultar lento y difícil de controlar por el hecho de producirse una única actualización de los parámetros hasta la convergencia de la función.

La expresión que representa el Descenso por Gradiente es la que se muestra en la ecuación 3.4

$$\Theta = \Theta - \alpha \cdot \nabla J(\Theta) \tag{3.4}$$

Donde α representaría el valor del *learning rate* y $\nabla J(\Theta)$ el gradiente de la función de pérdidas $J(\Theta)$.

De esta forma, SGD como variante, se encarga de llevar a cabo el cálculo del gradiente, pero en este caso a partir de grupos pequeños formados por datos del conjunto de entrenamiento en lugar de con todo el conjunto como se hace con G. Estos grupos pequeños de datos de entrenamiento son conocidos comúnmente con el nombre de *mini-batch* ya que están formados dentro de cada uno de los *batch* en los que se divide una época. Este hecho implica que a la hora de actualizar los parámetros internos del modelo, esta actualización se haga por cada grupo de datos. El hecho de que el conjunto de datos sea dividido en subconjuntos y que se produzcan tantas actualizaciones como subconjuntos puede hacer pensar que el proceso tardaría en completarse más tiempo que el empleado en GD, pero esto no es así, pues SGD proporciona una ejecución en un tiempo más reducido.

De esta forma, la expresión que representaría el Descenso por Gradiente Estocástico sería la que se muestra a continuación en la siguiente ecuación 3.5

$$\Theta = \Theta - \alpha \cdot \nabla J(\Theta; x(i); y(i)) \tag{3.5}$$

Donde, en este caso, $x(i)$ e $y(i)$ representarían los diferentes subconjuntos de datos de entrenamiento, siendo el resto de variables, las mismas que para el Descenso por Gradiente.

En cuanto a las ventajas que implica el uso de SGD frente a GD, se puede encontrar la ya comentada rapidez de ejecución junto con el hecho de que al producirse actualizaciones para cada subconjunto de datos, hace que se puedan encontrar diferentes valores que pueden hacer mínima la función permitiendo así poder encontrar el mejor de todos ellos.

Adam: El segundo de los optimizadores que se empleará en alguna parte del entrenamiento es Adam, el cual se encuentra detallado en [38]. Básicamente este algoritmo de optimización se encontraría dentro de la primera categoría mencionada anteriormente relativa a todos aquellos métodos que hacen uso de la derivada de primer orden para saber si la función de pérdidas se está acercando o alejando del mínimo. Adam, es considerado como una variante de SGD en el cual se produce una variación del *learning rate* a lo largo del entrenamiento de tal forma que permita ir adaptando su comportamiento a la búsqueda de ese mínimo. Es por eso que este optimizador es considerado como dentro del tipo adaptativo.

En cuanto a los valores fijados en este caso, se ha establecido dos posibles valores para el *learning rate* de Adam. Este valor dependerá de si se está realizando el entrenamiento desde cero con pesos aleatorios como ocurre en el caso de los *challenges* de carácter público donde se establecerá $\alpha = 10^{-3}$. El segundo valor que puede tomar dependerá de si se está realizando *Fine Tuning*. En este caso, como se detallará en la siguiente sección, cuando se parte de unos pesos obtenidos en un entrenamiento previo a partir de un conjunto de datos diferente, el valor adecuado para el *learning rate* en este caso sería algo inferior al fijado para la obtención de esos pesos de partida. De esta forma, en el caso de que se emplee *Fine Tuning* como en el conjunto de datos de hígado, el *learning rate* se verá disminuido una décima parte con respecto al valor de origen.

En cuanto al resto de parámetros propios de este optimizador, serán fijados con los valores recomendados para ello en el paper original, [38], siendo estos: $\beta_1 = 0{,}9$, $\beta_2 = 0{,}99$ y $\epsilon = 10^{-8}$

3.4.5 *Fine Tuning*

Otra de las estrategias muy común a la hora de entrenar una determinada red convolucional es no entrenarla desde cero. Este hecho implica tener previo entrenamiento, los pesos obtenidos para un conjunto de datos suficiente grande obtenido tras el entrenamiento de una determinada arquitectura. De tal forma

que partiendo de esa información, se transfieran esos pesos y se parta de ellos para entrenar la misma arquitectura sobre un conjunto de datos más pequeño y con menor variedad, modificando la red únicamente en las capas que se vean afectadas por determinadas características de los diferentes conjuntos de datos como puede ser el número de clases de cada uno de ellos. Este procedimiento es comúnmente conocido como *Fine Tuning* y aunque en el estudio realizado a lo largo de este documento no es aplicado en todos los conjuntos de datos, sí que se emplea en el entrenamiento relacionado con la base de datos propia al tratarse ésta de un *set* de datos de pequeñas dimensiones.

Aun así, *Fine Tuning* no es la única estrategia disponible para la transferencia de aprendizaje, existen otras opciones como puede ser el hecho de extraer características fijas que luego serían compartidas por otra arquitectura diferente a la original o mediante el empleo de modelos pre-entrenados, muy utilizados últimamente con el objetivo de adelantar el tiempo de entrenamiento. [39]

Debido a las características que posee el conjunto de datos de hígados humanos y al tratarse de un número muy reducido de *frames*, se ha optado por realizar los entrenamientos en las diferentes arquitecturas probadas para este caso a partir de los pesos obtenidos en todas ellas para el *challenge* público de instrumental más órganos en operaciones de cerdos. En este caso además se suma el hecho de que anatómicamente, el cerdo es uno de los animales que más se parece al ser humano por lo que en cuanto a apariencia física de los órganos, habría menos diferencias que en otros casos, pudiéndose aprovechar esta ventaja para obtener unos mejores resultados tras el *Fine Tuning* en el conjunto de datos de hígados.

Como detalle final, es importante mencionar que a la hora de afinar los pesos a partir de otros pesos obtenidos previamente, es muy común el hecho de establecer un *learning rate* o tasa de aprendizaje por debajo de la fijada para los pesos a partir de los cuales se hace *Fine Tuning*. Este hecho se debe principalmente a que se espera obtener unos resultados relativamente buenos por lo que es conveniente que éstos sean obtenidos de forma más lenta, pudiéndose evitar, de esta forma, *overfitting* en la red.

3.5 Detalles del entrenamiento

Tras conocer algunas de las estrategias seguidas a la hora de llevar a cabo los diferentes entrenamiento y a fin de conseguir los mejores resultados posibles en cuanto a la segmentación de los diferentes conjuntos de datos es importante tomar una buena decisión a cerca de determinados parámetros claves en la fase de entrenamiento como puede ser el valor del *Learning Rate*, el número de épocas empleadas o el tamaño de los *batch* entre otros. Para ello, a lo largo de esta sección se detallarán los aspectos más importantes a tener en cuenta en el entrenamiento de los diferentes conjuntos de datos, los cuales, recordemos que no tenían el mismo número de *frames* totales. De esta forma, en base al *challenge* utilizado, las características del entrenamiento serán las detalladas a continuación en la tabla 3.3:

Challenge	***Learning Rate***	***Tamaño del batch***	***Optimizador***	***Fine Tuning***
Instrumentos	10^{-3}	5	*Adam-SGD*	No
Instrumentos y órganos	10^{-3}	10	*Adam-SGD*	No
Hígados humanos	10^{-3}	5	*Adam-SGD*	Sí
Épocas: 400				

Tabla 3.3: Características de los entrenamientos.

De esta forma, como se puede observar en la tabla, en todos los casos el valor del **learnig rate** fijado al inicio del entrenamiento es de 10^{-3}. Este parámetro se encargará de indicar al optimizador empleado

durante el entrenamiento, qué tan lejos debe mover los pesos en la dirección opuesta al gradiente de cada uno de los subconjuntos de datos o *mini-batch*, por lo que la elección de un valor apropiado para el learning reta es de vital importancia ya que un valor demasiado alto puede hacer que el entrenamiento no converja o incluso llegue a divergir provocado por los cambios de peso tan grandes que se pueden producir, haciendo de esta forma que el mínimo de la función no se alcance o se sobrepase. Por el contrario, un valor de *learning rate* demasiado pequeño proporciona unos resultados más confiables añadiendo el impedimento de que el tiempo empleado hasta completar el entrenamiento sea muy largo ya que los pasos realizados hasta llegar al mínimo de la función son demasiado pequeños [40].

Para ver la influencia en la elección del valor del *learning rate*, se puede hacer referencia a una de las gráficas más significativas para este parámetro en la cual se ve el efecto producido para diferentes valores de *learning rate* en cuanto a la obtención de la función de pérdidas a lo largo de las épocas de dicho entrenamiento.

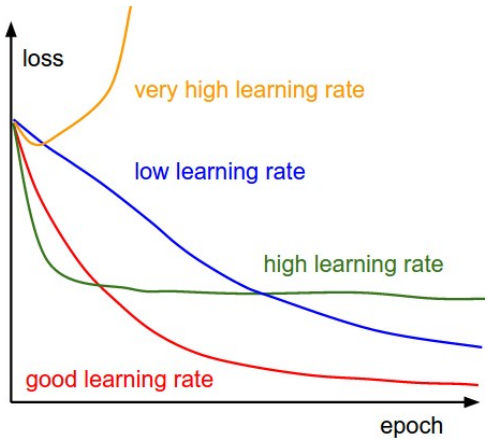


Figura 3.9: Efectos del *Learning Rate*

Por último, comentar que en lo que respecta al valor del *learning rate*, este se verá reducido en base al empleo de más de un optimizador así como al uso de *Fine Tuning*. Reduciéndose su valor una décima parte en cada caso.

Por su parte, en lo que a **Épocas** respecta, hay que mencionar que este parámetro ha sido fijado en un principio en 400 como valor aproximado pero que a lo largo de los diferentes entrenamientos, se ha hecho uso de las conocidas *callbacks* de Keras, las cuales te permiten obtener información a cerca del estado del entrenamiento. De esta forma, empleando la allbacks, *EarlyStopping*, se puede detener el entrenamiento cuando el valor monitorizado, no haya mejorado en un número determinado de épocas. Existe además *ModelCheckpoint* el cuál te permite guardar el último mejor modelo obtenido antes de que se produjera la detección de dicho entrenamiento. [41]

En lo que respecta al tamaño del **batch**, la estrategia seguida ha sido muy similar a la del número de épocas, donde dicho valor también se ha visto afectado por el número de *frames* que contienen cada base de datos para la fase de entrenamiento. De esta forma, no se ha encontrado ninguna documentación al respecto donde se especifique un valor correcto para cada caso, por lo que se ha optado por fijar un tamaño de 5 para las dos bases de datos pequeñas, y de 10 para el caso de instrumental y órganos. En cuanto a la determinación de esos dos valores concretamente, decir que han sido elegidos por ser dos valores muy comunes.

Por su parte, en cuanto al uso de optimizadores, en todos los casos se ha seguido la técnica de emplear primero Adam y posteriormente SGD. El motivo de por qué se han empleado conjuntamente y en este orden es debido a que, según se detalla en [38], Adam proporciona en las primeras épocas, una caída

más drástica hacia el mínimo de la función que SGD, para posteriormente, en la mayoría de los casos, quedarse oscilando en torno a valores muy próximos sin sufrir variaciones demasiado notables. Es en ese momento cuando se emplearía como optimizador SGD a fin de salir de esa zona de oscilación en busca de un mínimo más pequeño aún.

Por último, en lo relativo al empleo o no de *Fine Tuning*, en este caso solo se ha hecho uso de esta técnica para llevar a cabo el entrenamiento de la base de datos de hígados humanos por ser de un parecido razonable en cuanto a los *frames*, con la base de datos de intervenciones porcinas. Para mayor detalle, recurrir de nuevo a la sección 3.4.5.

Finalmente, comentar que en las bases de datos en las que no se dispone de un *set* de datos específico para la fase de validación, éste ha sido separado de forma automática durante el entrenamiento en algunos casos y de forma manual en otros. En lo que respecta a los *challenges* públicos, los *frames* destinados a la fase de validación son separados dentro del proceso de validación, destinando un 10 (%) aleatorio del conjunto de entrenamiento para usarlo como validación al finalizar cada época. Por su parte, en cuanto a la base de datos propia, recordar que el conjunto de validación fue separado de forma manual en función de la similitud entre *frames* y a fin de garantizar que fueran lo más diferentes posibles. Para mayor detalle, recurrir a la sección 3.3.2.

3.6 Métrica empleada: IoU

Por último, en cuanto a la obtención de los diferentes resultados, obtenidos tras la fase de entrenamiento, es necesario poder determinar cuánto de buenos son estos resultados para, posteriormente llevar a cabo su correspondientes comparativa entre las diferentes arquitecturas probadas y detallarlas en el capítulo, 5. Es por eso que una comparativa debe estar fundada a partir de una métrica que permita igualar los resultados de alguna forma, pudiendo así determinar los aspectos más generales. Para ello, en este caso se llevará a cabo el cálculo del Intersection Over Union (IoU) así como del *mean-IoU* detallados ambos a continuación.

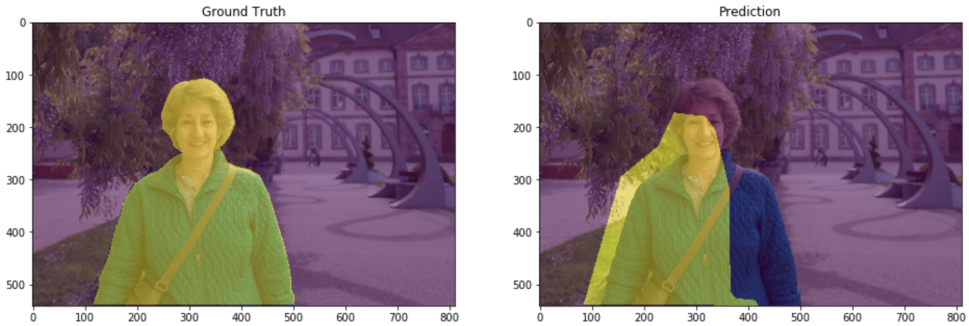
La intersección sobre la unión, es una de las métricas más empleadas en el campo de la segmentación ya que es capaz de proporcionar información relativa a cuánto de buena es la segmentación obtenida en comparación con el correspondiente *ground-truth* de la imagen. De esta forma, el IoU obtiene una medida de la similitud que existe entre la predicción de una determinada región en un frame de *test* frente al *ground-truth* de esa misma región [42]. Así, la intersección sobre la unión quedaría definida mediante la expresión 3.6:

$$IoU = \frac{TP}{FP + TP + FN} \tag{3.6}$$

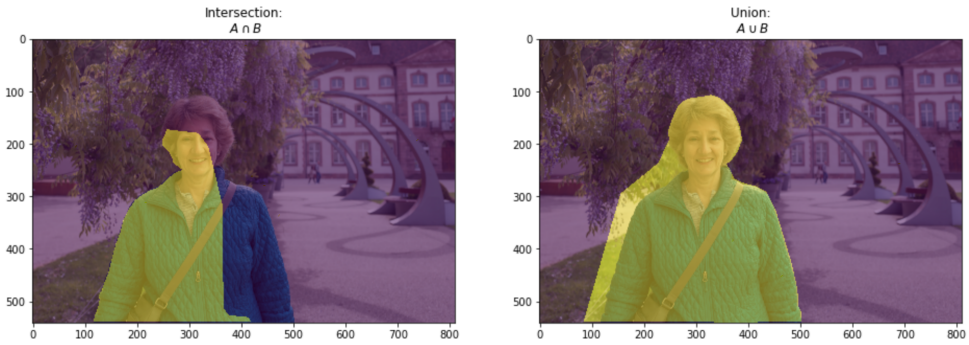
Donde TP representaría los aciertos realizados durante la segmentación comparando el resultado con el GT. Mientras que FP y FN representaría los falsos positivos y falsos negativos respectivamente.

De esta forma, a partir de la expresión 3.6, el IoU también se podría representar como el cociente de la intersección entre la predicción y el *ground-truth* con la unión de ambas, AND y OR respectivamente. Pudiéndose entonces, expresar también el IoU como:

$$IoU = \frac{GT \cap prediction}{GT \cup prediction} \tag{3.7}$$



(a) Groundt Truth y Predicción



(b) Intersección y Unión

Figura 3.10: Cálculo del IoU, [13]

Para ver con más claridad la métrica obtenida a través del IoU, se proporciona a continuación un ejemplo visual en el que se aprecia como, a partir del groundt truth y la predicción, se calcula la unión y la intersección entre ambos [13].

En cuanto a la determinación de cuándo la intersección sobre la unión proporciona un resultado considerado como bueno, normalmente se establece que todos aquellos valores que se encuentren por encima del 0.5, siendo por lo tanto, la similitud entre la imagen segmentada y el *ground-truth* superior al 50(%), se puede decir que esa predicción es considerada como una buena predicción [42], aunque sin duda cuanto más cercada sea a la unidad, mayor similitud entre ellas habrá.

De esta forma, mediante el empleo del IoU se puede conocer la similitud en una región determinada de la imagen con respecto al GT de la misma, pero si se quiere saber cuánto de buena es la segmentación obtenida con respeto a todas las regiones o clases disponibles en la imagen, habría que emplear *mean-IoU*. Donde, una vez calculado el IoU de cada una de las clases que aparecen en la imagen, se llevaría a cabo la media de los IoU calculados para ese caso teniendo en cuenta el número total de clases que aparecen en esa imagen en concreto [13].

3.7 Conclusiones

A lo largo de este capítulo se ha intentado dar todos los detalles posibles a la forma en la que se ha abordado el problema de la segmentación en imágenes médicas. De esta forma, partiendo de los conceptos previos desarrollados en el capítulo 2, se ha especificado todas las herramientas necesarias para dicho estudio.

Así, partiendo de la arquitectura propuesta, se ha podido ver cuál es el esquema que ésta sigue así como las diferentes capas que la forman hasta la obtención de la imagen de salida en la que se encontrarían

las diferentes clases segmentadas, pudiéndose ver en detalle también la función de pérdidas empleada para conseguir dicho objetivo.

Conocida la estructura de red a emplear, se procedió a detallar las diferentes bases de datos que se iban a emplear en las diferentes fases de entrenamiento, validación y *test*. Pudiendo conocer los detalles de cada una de ellas, tales como las dimensiones de las imágenes que forman dichos conjuntos de datos, la procedencia de cada una de ellas así como el fin buscado en cuanto a segmentación. Detallando en profundidad los casos en los que ha sido necesario realizar una etiquetación manual para poder emplear esas imágenes como prueba.

Tras esto, se han detallado las estrategias seguidas a la hora de llevar a cabo la fase de entrenamiento, donde ha sido de vital importancia el ajuste de clases realizado, así como el aumento de datos debido a la poca variedad en las imágenes de las que se disponía. Con esto, la elección de los optimizadores empleados dentro de los posible que hay, también ha sido un detalle a tener en cuenta pues ha ayudado a lo largo del entrenamiento a conseguir unas tasas de accuracy bastante cercanas a 1. Por último, dentro de las estrategias empleadas, se detalló también el uso de técnicas como *fine tuning* que dan la posibilidad de entrenar un determinado conjunto de datos a partir de los pesos conseguidos en la misma arquitectura pero para otra base de datos diferente.

Conocidas las estrategias a emplear, se detallaron por último, todos los parámetros más significativos del entrenamiento así como la métrica empleada en la toma de resultados a fin de poder conseguir unos datos que permitan comparar los resultados obtenidos en cada una de las diferentes pruebas realizadas y poder sacar así unas conclusiones.

De esta forma, tras la finalización de este capítulo, se da paso al capítulo 4 donde se mostrarán los diferentes resultados obtenidos en cada una de las pruebas realizadas.

Capítulo 4

Resultados

No hay fracasos, solo lecciones por aprender.

Oprah Winfrey

4.1 Introducción

Tras describir el modelado del problema, así como los procedimientos seguidos para realizar la fase de entrenamiento, se describen a continuación los diferentes resultados experimentales obtenidos. Se han utilizado las bases de datos disponibles de segmentación semántica para entrenar y evaluar la propuesta de red para este trabajo, así como las arquitecturas UNet y SegNet, muy significativas en el campo de la segmentación semántica de imágenes médicas. La red SegNet utilizada consistirá en el modelo *basic* proporcionado por los creadores.

De esta forma, este capítulo de resultados se organizará en base a los diferentes conjuntos de datos utilizados para la toma de resultados, organizándose así en tres grandes secciones correspondientes a cada una de las bases de datos empleadas. Para cada una ellas se mostrarán los resultados obtenidos por cada arquitectura de red así como las métricas obtenidas en cada uno de los casos.

4.2 *Challenge*: Instrumental y órganos en intervenciones porcinas

Este *challenge*, como se ha desarrollado en la sección del capítulo 3 y concretamente en 3.3.1.2, es de carácter público. Se divide de forma original en datos de *train* y de *test*, siendo cada uno de dimensiones 2235 y 1000 *frames* respectivamente. En este caso, el número total de clases son 12 enumeradas de la 0 a la 11.

Debido a que no se dispone de los *ground truth* del conjunto de *test*, se ha optado por llevar a cabo una segunda división que permita disponer de un conjunto de datos de *train*, validación y *test* separadas en base a un 80(%), 10(%) y 10(%) respectivamente del total de imágenes del conjunto de entrenamiento original. Se garantiza así que para la parte de *test* se dispone de los *ground truth* de cada una de las imágenes probadas. De esta manera, se puede realizar el cálculo de la métrica indicada en la sección 3.6 del capítulo 3. En todo caso y de manera cualitativa, se mostrarán algunos resultados de segmentación obtenidos con las imágenes de *test* originales.

La partición de los datos disponibles en conjuntos de *train*, validación y *test* se realiza de forma aleatoria. Se dispone así de un total de 1811 *frames* para *train*, 201 para validación así como 223 para *test*, además de disponerse de las imágenes de *test* originales sin *ground truth* que los creadores aportaban con el *challenge* completo.

Tal y como se ha detallado en el capítulo anterior, en el entrenamiento se emplearán los optimizadores Adam y SGD. Se establece un total de 400 épocas, tamaño de *batch* de 10 y Adam como optimizador inicial con *learning rate* de 10^{-3}, siendo el resto de parámetros los recomendados en [38], esto es $\beta_1 = 0{,}9$, $\beta_2 = 0{,}99$ y $\epsilon = 10^{-8}$. El conjunto de validación se utiliza para seleccionar en que *epoch* se ha encontrado el mejor modelo de red y para condicionar la parada del proceso de entrenamiento. De esta forma, tras estancarse dicha mejora y activarse por lo tanto la condición de parada fijada, se partirá de los mejores pesos almacenados para comenzar un segundo entrenamiento en el que se empleará SGD como optimizador con *learning rate* de 10^{-4}, a fin de salir de la zona de oscilación creada con el final del anterior entrenamiento.

Los resultados obtenidos por las tres arquitecturas de red propuestas se muestran en la tabla 4.1, donde se indica la medida del IoU medio de todas las clases en el conjunto de *test*.

IoU	UNet	SegNet	MIS-Net
Clase 1	86.60(%)	86.25(%)	86.60(%)
Clase 2	72.91(%)	72.53(%)	71.75(%)
Clase 3	60.43(%)	62.46(%)	63.06(%)
Clase 4	85.83(%)	85.44(%)	86.11(%)
Clase 5	85.94(%)	86.30(%)	86.30(%)
Clase 6	50.55(%)	47.56(%)	39.36(%)
Clase 7	70.53(%)	71.41(%)	63.17(%)
Clase 8	0.00(%)	0.00(%)	0.00(%)
Clase 9	58.75(%)	59.02(%)	61.97(%)
Clase 10	85.11(%)	83.14(%)	83.93(%)
Clase 11	72.77(%)	67.55(%)	74.52(%)
medio	76.91(%)	76.52(%)	76.05(%)

Tabla 4.1: Resultados de IOU medios en el conjunto de *test*

En la figura 4.1 se muestran los resultados individuales de la segmentación obtenida en 8 ejemplos escogidos del conjunto de *test*. Así mismo, en la tabla 4.2, se muestran los valores de IoU obtenidos para cada uno de los ejemplos mostrados en la figura 4.1, proporcionando tanto el IoU por clase como el medio de cada uno de los diferentes casos de análisis.

Como se observa en los resultados aportados, MIS-Net obtienen una segmentación muy próxima a la de la UNet y la SegNet tanto a nivel de clases como de media. En cuanto a detalles significativos, cabe mencionar la clase 8, correspondiente con las agujas de sutura. En este caso, esta clase aparece una sola vez en el conjunto de *test* y como se puede observar en la tabla 4.1, ninguna de las redes es capaz de detectarla. Con respecto al resto, se puede ver como los mejores resultados se encuentra en la clase 1 la cual representa el eje del instrumento, así como en la 4,5 y 10, representando la parénquima renal, la cubierta del riñón y el intestino delgado respectivamente. De esta forma, conociendo el tipo de elemento a segmentar se puede observar cómo aquellos que tienen unas dimensiones considerables dentro de la escena, como son los órganos o una forma muy determinada como es el eje del instrumento son segmentados en todos los casos con una mejor calidad que el resto de clases.

Además, es importante comentar que debido a que los datos de *test* están sacados del conjunto de entrenamiento original, las métricas obtenidas serán mucho mejores que en el caso de probar con unos *frames* con ligeras diferencias, donde los resultados empeorarán notablemente. Este caso se puede

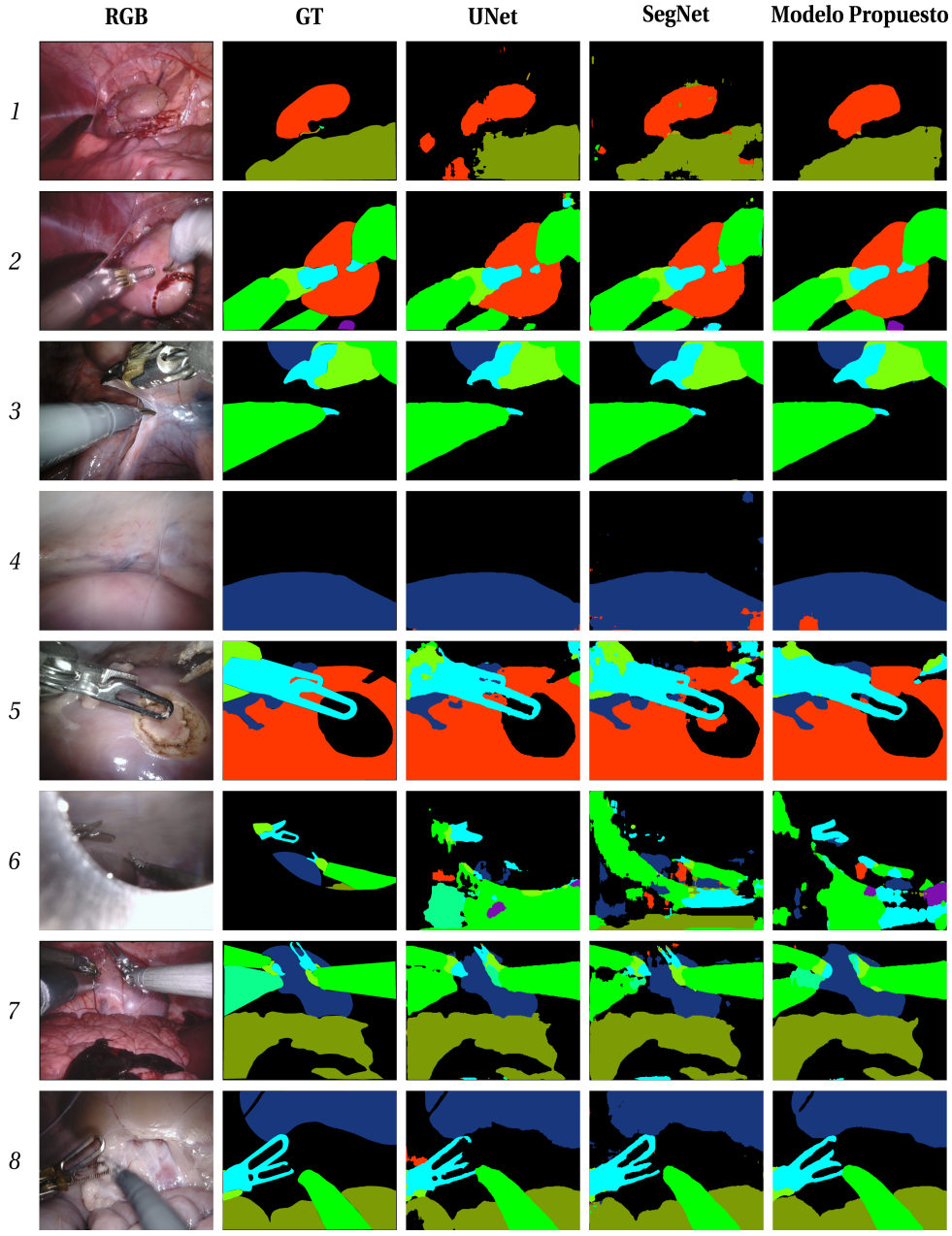


Figura 4.1: Comparativa de resultados obtenidos

		Clases										
Ej.	**Red**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	**UNet**	-	-	-	66.23	-	14.22	0.00	-	-	77.78	-
	SegNet	-	-	-	83.93	-	31.43	0.00	-	-	80.64	-
	MIS-Net	-	-	-	85.89	-	14.28	0.00	-	-	88.60	-
2	**UNet**	89.42	76.61	68.10	91.09	-	-	-	-	2.88	-	-
	SegNet	89.35	76.81	72.26	91.87	-	-	-	-	5.73	-	-
	MIS-Net	88.04	83.07	72.84	91.74	-	-	-	-	62.69	-	-
3	**UNet**	96.61	91.74	93.04	-	94.85	-	-	-	-	-	-
	SegNet	96.51	91.20	92.96	-	95.17	-	-	-	-	-	-
	MIS-Net	96.80	90.82	93.00	-	92.87	-	-	-	-	-	-
4	**UNet**	-	-	-	-	97.69	-	-	-	-	-	-
	SegNet	-	-	-	-	93.37	-	-	-	-	-	-
	MIS-Net	-	-	-	-	95.75	-	-	-	-	-	-
5	**UNet**	-	76.96	28.02	94.12	74.35	-	-	-	-	-	-
	SegNet	-	72.09	42.33	89.91	68.17	-	-	-	-	-	-
	MIS-Net	-	77.91	29.37	93.51	71.18	-	-	-	-	-	-
6	**UNet**	0.17	35.66	4.65	-	19.97	-	-	-	-	0.00	-
	SegNet	14.65	5.22	21.59	-	21.47	-	-	-	-	1.69	-
	MIS-Net	7.09	14.28	4.51	-	0.00	-	-	-	-	0.22	-
7	**UNet**	59.29	31.39	11.19	-	76.43	-	-	-	-	94.38	0.00
	SegNet	65.70	37.60	42.03	-	72.83	-	-	-	-	92.27	8.93
	MIS-Net	67.26	21.83	26.78	-	81.40	-	-	-	-	91.28	24.54
8	**UNet**	87.23	79.99	60.13	-	93.65	-	-	-	-	87.34	-
	SegNet	89.30	80.53	23.66	-	91.36	-	-	-	-	92.09	-
	MIS-Net	86.72	74.88	61.34	-	92.84	-	-	-	-	93.46	-

Tabla 4.2: IoU obtenidos

contemplar en la secuencias de imágenes de la figura 4.2, donde se prueba el conjunto de *test* original proporcionado por los autores sobre las diferentes arquitecturas.

Como en este caso no se dispone de los *ground truth* como bien se ha indicado, la comparativa sería visual, observándose que la segmentación obtenida para las tres arquitecturas empeora ligeramente en comparación con los resultados mostrados para los datos de la figura 4.1.

Aquí, al igual que en los ejemplos anteriores, se puede decir que en base al tipo de escena analizada, los resultados serán mejores o peores ya que como se puede observar, en casos como el ejemplo 1 o 2, al encontrarse el órgano reconocibles así como los instrumentos en una posición fácil de detectar, la segmentación obtenida es mucho mejor que la que se puede aportar para casos como el ejemplo 6, 7 u 8, donde lo único que las tres redes consiguen llegar a reconocer de mejor forma son los instrumentos empleados en la intervención.

A rasgos generales, aunque los resultados parecen ser de una calidad inferior a los mostrados en 4.1, se puede decir que MIS-Net para este estudio tendría un comportamiento similar a la UNet, red que se caracteriza por ser una de las mejores dentro de la detección y segmentación en imágenes médicas. Pudiéndose añadir además que en algunos casos como en el ejemplo 1 o 2, la segmentación proporcionada por MIS-Net, alcanza una mejor segmentación de la imagen que la que es capaz de proporcionar UNet.

Como bien se ha podido observar, los resultados obtenidos en ambas figuras, 4.1 y 4.2 son muy diferentes entre sí. Este hecho se debe básicamente a que los datos correspondientes a la figura 4.2 contemplan intervenciones distintas a las que se incluyen en los datos iniciales. Esta diferencia conlleva a que las arquitecturas probadas no sean capaces de segmentar de la misma forma las imágenes *testeadas*. Lamentablemente este problema estará presente en el resto de conjuntos de datos donde la mínima variación en la escena analizada puede desembocar en unos resultados con una segmentación de peor calidad. Aún así, la variabilidad de la imagen es un aspecto que habrá que afrontar pues al tratarse de intervenciones quirúrgicas no se garantiza la igualdad en el interior de cada uno de los pacientes. De esta forma, el principal problema que se puede mencionar es que para obtener unos resultados de calidad en temas médicos, habrá que disponer de una cantidad muy elevada de imágenes a fin de que las redes

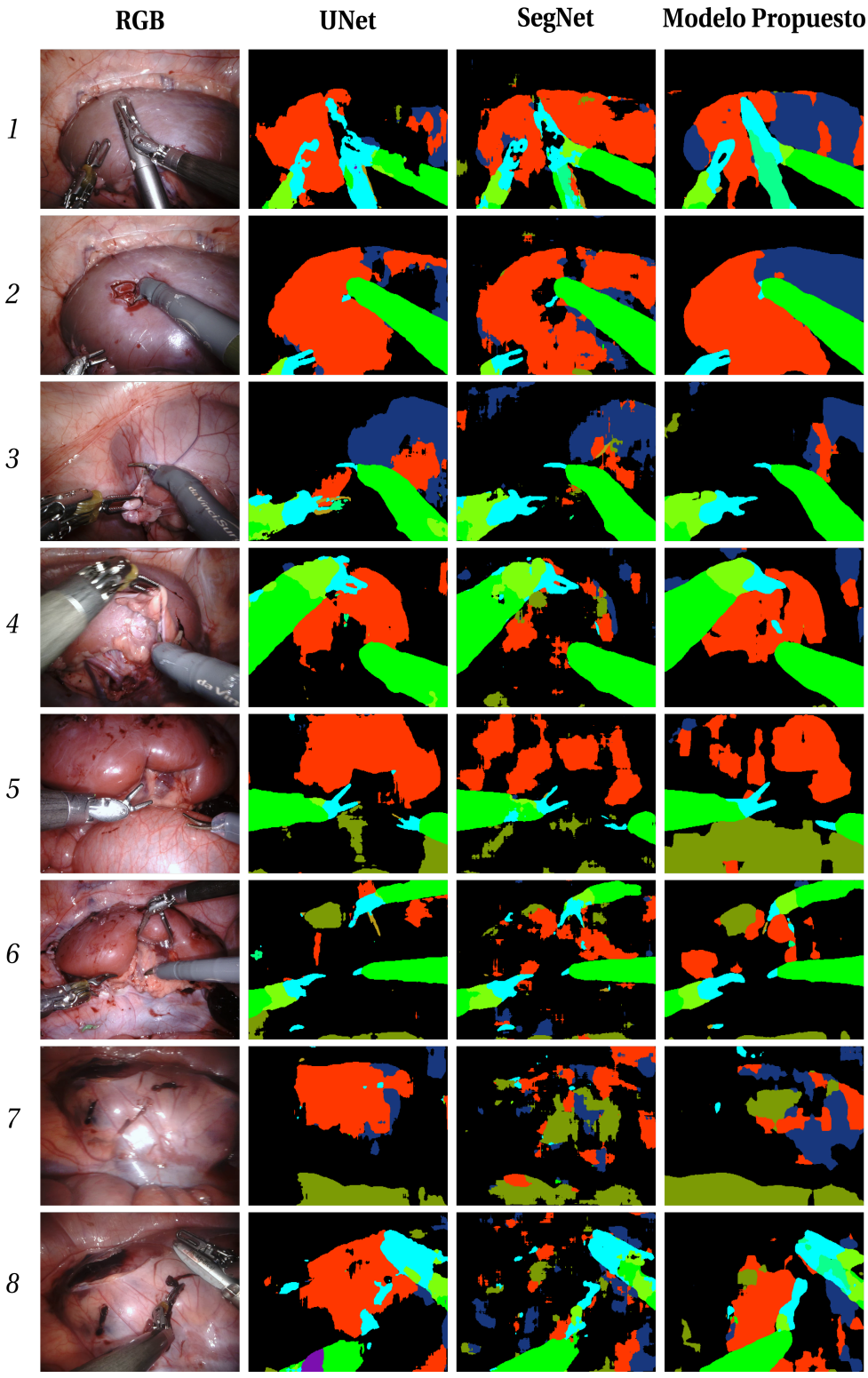


Figura 4.2: Resultados imágenes de *test*

	UNet	SegNet	Modelo Propio
Tiempo de predicción (ms)	17.2	28.9	27.6
FPS	58.14	34.6	36.23

Tabla 4.3: Tiempos de ejecución base de datos intervenciones porcinas

vean un gran número de imágenes de cada clase. Pues a nada que aparezca una secuencia con ligeras diferencias, la red tendrá problemas para segmentarla.

Una vez concluida la fase de entrenamiento se llevó a cabo la predicción de cada uno de los resultados mostrados anteriormente, donde es importante conocer el tiempo transcurrido hasta la obtención de cada una de las predicciones. De esta forma, en la tabla 4.3 se pueden ver el tiempo en media que necesita cada red hasta conocer los resultados del conjunto entero de *test*.

Teniendo en cuenta que a partir de 20 FPS se podría considerar que una determinada red trabaja en tiempo real, como se puede observar en la tabla 4.3, las tres redes empleadas en la comparativa superan este valor estando la UNet muy por encima. En este caso, el modelo propuesto tendría un tiempo de ejecución similar al de la SegNet pero ligeramente superior.

Por último, comentar que en lo que respecta al entrenamiento empleado para cada base de datos, en todos los casos será el mismo que el descrito en esta sección, incluyendo el uso de optimizadores y parámetros empleados, por lo que en los sucesivos apartados no se detallará nuevamente.

4.3 *Challenge*: Instrumental

Por su parte, en lo que respeta a la base de datos de instrumento, cuyo objetivo es la detección de las dos partes que conformarían dicho instrumento: eje y mango, no se dispone tampoco de los *ground truth* de las imágenes originales de *test*. De esta forma, se ha optado por seguir un procedimiento similar al realizado en el caso anterior, donde del total de 160 *frames* destinados a la fase de *train*, han sido separados el 10(%) para validación así como el mismo porcentaje para *test*. Con esto, se ha obtenido una división de 128 imágenes para *train* y 16 tanto para validación como para *test*.

Así, tras disponer de un *set* de *test* que dispone de los correspondientes GT de cada una de las imágenes, se puede llevar a cabo el cálculo de los diferentes IoU de cada clase así como el valor medio que contemplaría el total de clases. Mostrándose en la tabla 4.4 los resultados obtenidos para ambos casos del conjunto total de *test*.

IoU	UNet	SegNet	MIS-Net
Clase 1	34.83(%)	42.68(%)	38.90(%)
Clase 2	52.68(%)	55.45(%)	54.44(%)
medio	44.56(%)	49.71(%)	47.52(%)

Tabla 4.4: Comparativas de IoU obtenido

Así mismo, en la figura 4.3 se pueden contemplar los resultados individuales de la segmentación obtenida para 4 ejemplos escogidos al azar dentro del conjunto de *test*. En cuanto a sus valores de IoU por clase, éstos son mostrados en la tabla 4.5.

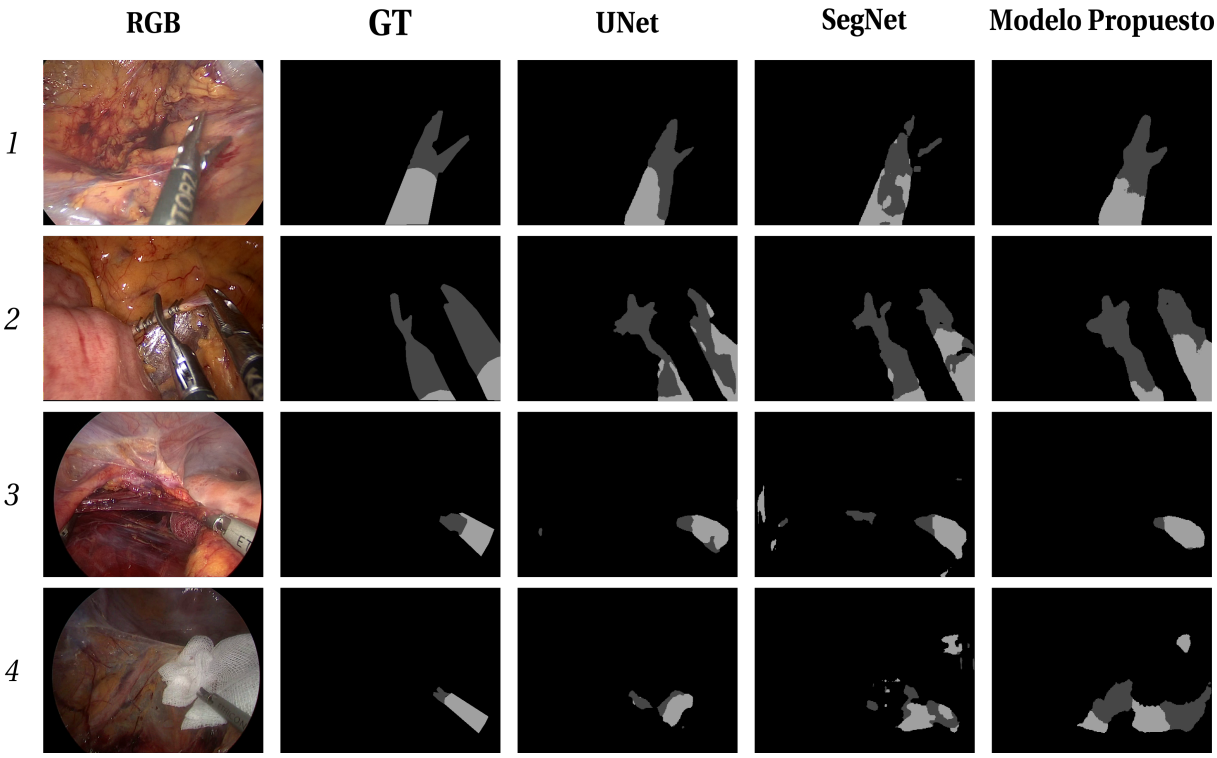


Figura 4.3: Comparativa de resultados obtenidos

		Clases	
Ej.	**Red**	**1**	**2**
1	**UNet**	47.04	66.85
	SegNet	40.12	51.10
	MIS-Net	55.64	68.63
2	**UNet**	43.11	28.75
	SegNet	60.62	52.47
	MIS-Net	51.06	45.13
3	**UNet**	32.09	60.44
	SegNet	33.64	60.3
	MIS-Net	27.42	63.38
4	**UNet**	12.35	10.62
	SegNet	9.15	11.60
	MIS-Net	0.00	0.08

Tabla 4.5: IoU obtenidos

En cuanto a los resultados generales obtenidos para cada una de las redes, se puede decir que en este caso las tres arquitecturas tendrían un comportamiento similar frente a esta base de datos. Aún así, es importante destacar la complejidad de la escena, pues dependiendo de ésta, la segmentación obtenida puede ser mejor o peor. Este detalle se puede apreciar en los dos primeras ejemplos mostrados en la figura 4.3. En estas escenas, los elementos principales que intervienen son los instrumentos con sus diferentes partes bien diferenciadas, lo que facilita la segmentación a la hora de distinguir entre una clase u otra. Por el contrario, si la escena se caracteriza por una cierta dificultad al contener además de instrumental otro elemento de confusión como gasas, la segmentación obtenida estaría menos definida. Este ejemplo se puede apreciar en la escena 4 de la figura 4.3 donde además, al observar la tabla 4.5 se puede ver como posee los valores de IoU más bajos.

En lo que respecta al comportamiento que MIS-Net presenta en este conjunto de datos, se puede observar mediante la tabla 4.4 que los valores obtenidos tanto por clase como medios, son ligeramen-

te superiores a la UNet. Mientras que en lo que a la SegNet respecta, estarían muy a la par ambas arquitecturas.

Si además se lleva a cabo el mismo proceso de prueba pero dentro del *set* de datos de *test* originales del propio *challenge*, la calidad de los resultados empeoraría. Esto se puede observar en la figura 4.4 donde a medida que se complica la escena analizada, la segmentación empeora pues no se delimitan las clases con claridad. De nuevo, se cumple el hecho de que todas aquellas escenas sencillas con el instrumental diferenciado en la imagen, obtienen resultados superiores que si se incorporan elementos nuevos que confundan a la red. Esto se puede apreciar en la escena 4 de la figura 4.4 donde la presencia de una gasa, nuevamente vuelve a perjudicar la segmentación. Por su parte, casos como el mostrado en la escena 1, 5 o 6 demuestran el hecho de que cuanto más aislado esté el instrumental, mejor será la segmentación.

Nuevamente vuelve a aparecer el problema mencionado en la sección anterior, donde la variabilidad así como la cantidad de las imágenes, son dos factores que afectan notablemente a la segmentación. Demostrándose que al disponer de un conjunto limitado de imágenes, una mínima diferencia al tratarse de una operación diferente a las ya aprendidas, puede general problemas en la identificación de clases.

Al igual que en la base de datos anterior, tras finalizar el entrenamiento se procedió a la toma de los resultados expuestos donde los tiempos de ejecución obtenidos para cada una de las tres arquitecturas probadas han sido los mostrados en la tabla 4.6

	UNet	**SegNet**	**MIS-Net**
Tiempo de predicción (ms)	15.27	24.8	22.73
FPS	65.5	40.32	43.99

Tabla 4.6: Tiempos de ejecución de la base de datos propia

Como se puede observar en la tabla 4.6, al tratarse de una base de datos de menores dimensiones, en general el tiempo de ejecución de las diferentes redes es inferior al resto, estando la UNet en este caso también, muy por encima del resto.

4.4 *Challenge*: Intervenciones en hígados humanos

La peculiaridad de esta base de datos es que está formada a partir de *frames* sacados de diferentes intervenciones de laparoscopia sobre personas con algún problema de hígado. Como bien se ha comentado en la sección 3.3.2, el conjunto de imágenes ha sido etiquetado de forma manual, disponiéndose de esta forma, del GT de los *frames* que intervendrán en el proceso de entrenamiento. Este hecho hace que no sea necesario llevar a cabo ninguna separación extra, dividiendo así el total en conjunto de *train*, validación y *test* en base a una selección minuciosa en la que se ha intentado evitar la repetividad de varios *frames* consecutivos dentro del mismo conjunto de validación. En cuanto al conjunto de *test* está formado a partir de una selección aleatoria de imágenes, estando el resto de datos destinados para *train*.

De esta forma, manteniendo los parámetros descritos en la sección 4.2 se lleva a cabo la fase de entrenamiento. Donde la única diferencia con respecto a las anteriores reside en el empleo de fine tuning a partir de los pesos obtenidos con la base de datos de intervenciones porcinas. Es importante mencionar que de las 6 etiquetas de las que dispone la base de datos, solo se hará uso de las dos primeras clases por ser las más parecidas a las clases existentes en la base de datos de la cual se parte para hacer fine tuning. De esta forma, se buscarán las clases de hígado e instrumental considerándose el resto como 0.

Tras concluir con el entrenamiento se obtiene resultados como los mostrados en la tabla 4.7 donde se puede apreciar el IoU obtenido para cada clase así como el medio del conjunto de *test*. A su vez, en la

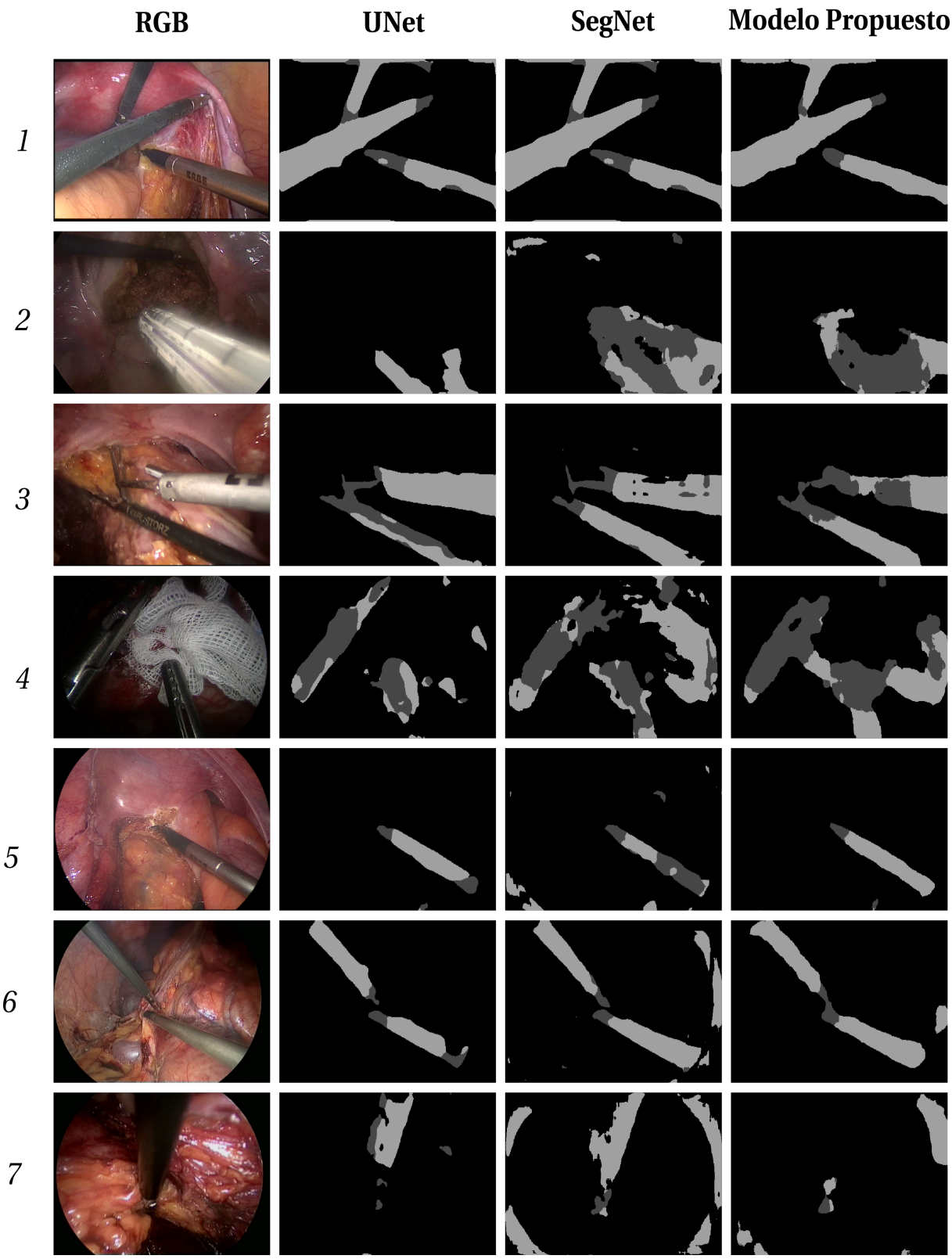


Figura 4.4: Resultados *set* datos de de *test*

figura 4.5 se puede apreciar una serie de ejemplos elegidos del conjunto de *test* donde se ve la segmentación obtenida para cada red.

IoU	UNet	SegNet	MIS-Net
Clase 1	87.02(%)	80.25(%)	88.29(%)
Clase 2	28.92(%)	27.39(%)	27.51(%)
medio	72.84(%)	67.96(%)	73.13(%)

Tabla 4.7: Comparativas de IoU obtenido

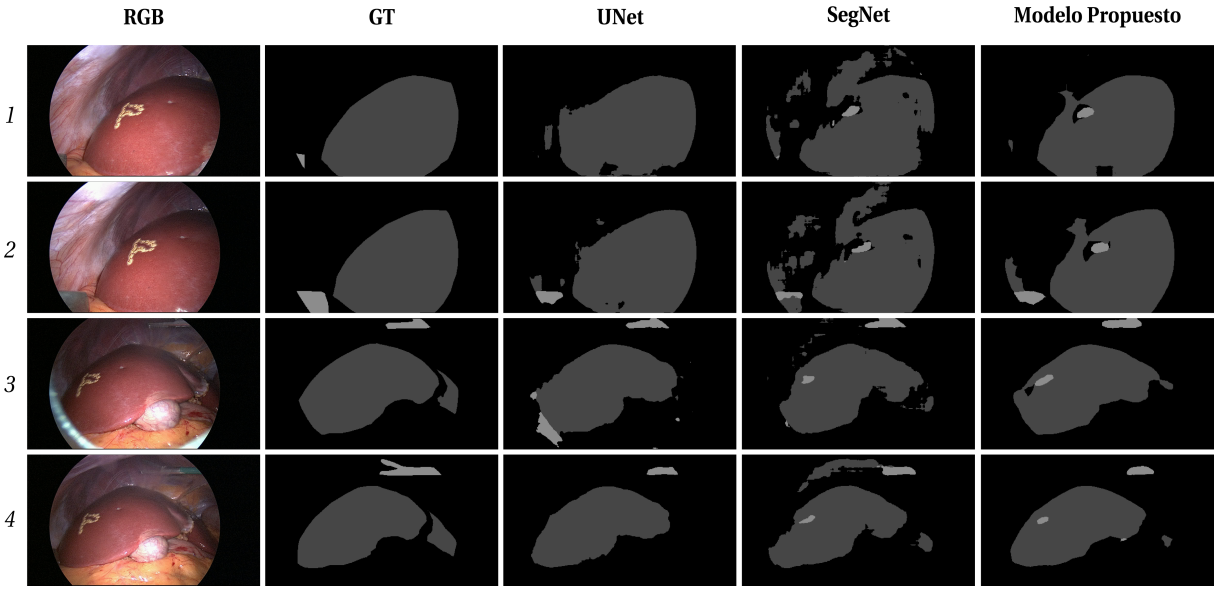


Figura 4.5: Resultados *set* datos de *test*

Donde, como bien se puede apreciar en la figura 4.5, aparentemente los resultados de peor calidad son los proporcionados por la SegNet, ya que a la hora de limitar las zonas de separación entre las diferentes regiones, lo hace peor que en el caso de la UNet o MIS-Net.

Por otro lado, otro detalle importante es la presencia de la *P* en el hígado del paciente. Donde como bien se puede apreciar, redes como la SegNet o MIS-Net, intentan segmentar esa p como si de un instrumento se tratara. Este hecho se debe a determinados factores como el brillo o el color que pueden confundirlas con la clase instrumental. Por su parte, la UNet en este sentido no falla y lo considera como una parte más de la clase hígado.

En cuanto a la clase instrumental, hay que destacar que en general en las tres arquitecturas es segmentada aunque no tan en detalle en comparación con el *ground truth*. Dentro de esta clase, en ocasiones se considera el tubo empleado para la inserción de los instrumentos así como de la cámara en el interior de paciente como una parte a segmentar confundiéndolo con la clase instrumental. Para mayor detalle, en la tabla 4.8 se pueden observar los IoU por clase obtenidos por cada arquitectura probada.

Centrando la atención en el IoU obtenido por clase mostrado en la tabla 4.8, se puede comprobar como la clase 1 perteneciente al hígado es donde mayor tasa de aciertos se produce al ser la superficie de mayor dimensiones. Por su parte, en las tres redes se cumple que la clase 2, perteneciente al instrumental empleado, es la que menor valor de IoU tiene, lo que indica que al ser superficies pequeñas y fáciles de confundir hace que sea la clase que mayor trabajo requiere

Como en el resto de conjunto de datos, se observa que al probarse con imágenes similares a las ya vistas por la red, los resultados obtenidos van a ser mejores que en el que caso de probar sobre imágenes

		Clases	
Ej.	**Red**	**1**	**2**
1	**UNet**	89.42	0.00
	SegNet	72.18	3.33
	MIS-Net	92.20	0.00
2	**UNet**	93.96	48.62
	SegNet	71.48	24.09
	MIS-Net	89.01	39.65
3	**UNet**	89.99	35.24
	SegNet	84.23	56.05
	MIS-Net	88.97	54.28
4	**UNet**	85.31	38.49
	SegNet	78.3	42.54
	MIS-Net	87.41	31.57

Tabla 4.8: IoU obtenidos

nuevas con ligeras diferencias. Pues en este caso, al igual que en los anteriores, se refuerza la idea de que en el momento en el que se disponga de una imagen perteneciente a una operación diferentes con un hígado desconocido, los resultados esperados serán peores.

Por último, en cuanto al tiempo transcurrido en las predicciones de cada una de las redes en las que se ha probado esta base aplicando fine tuning desde la base de datos porcina, se han podido obtener los tiempos ejecución mostrados en la tabla 4.9.

	UNet	**SegNet**	**Modelo Propio**
Tiempo de predicción (ms)	17.06	26.87	23.6
FPS	58.6	37.22	42.37

Tabla 4.9: Tiempos de ejecución base de datos propia

En este caso, la UNet sigue teniendo un tiempo de ejecución superior al resto a diferencia de que en concreto, en esta base de datos, el modelo propuesto es bastante superior a la SegNet.

Capítulo 5

Conclusiones y líneas futuras

En este apartado se resumen las conclusiones obtenidas y se proponen futuras líneas de investigación que se deriven del trabajo.

5.1 Conclusiones

El Trabajo de Fin de Máster desarrollado ha consistido en la creación y *testeo* de una red neuronal destinada a la segmentación semántica en imágenes médicas. Para ello, no solo se ha diseñado el esquema de dicha red conocida como MIS-Net sino que además se ha podido probar en tres bases de datos diferentes a fin de comparar los resultados obtenidos en cada una de ellas.

En lo que respecta a las bases de datos, cabe destacar que ha sido necesario unificar la nomenclatura empleada en el etiquetado de dos de ellas, catalogada como de carácter público al tratarse de *challenge*s utilizados para las distintas ediciones de MICCAI. El objetivo de estas bases de datos es la identificación de las distintas partes que forman el instrumental empleado en una operación así como la detección tanto de instrumental como de órganos en intervenciones porcinas. En cuanto al número de imágenes de las que se dispone, éste es variable pues una tiene mayor cantidad de muestras que la otra.

En cuanto a la tercera de las bases de datos, fue creada por el hospital *Chu Estaing* de Clermont-Ferrand en colaboración con el grupo *EnCoV* de la Universidad Clermont Auvergne en Francia. En ella se muestran diferentes operaciones de hígado a través de laparoscopia. La peculiaridad de este conjunto de datos es que ha sido etiquetado al completo para este trabajo, creando así 7 posibles clases a segmentar. El número de datos de los que se dispone en este caso es muy reducido, lo que ha propiciado el empleo de varios mecanismos de aumento de datos así como estrategias de entrenamiento tales como *fine tuning*.

Tras disponer de las diferentes bases de datos que iban a ser probadas, se llevó a cabo el entrenamiento. En él se fijaron una serie de parámetros comunes a todos los conjuntos de datos como fueron los optimizadores empleados, el valor del *learning rate* así como el número de épocas o el tamaño del *batch* utilizado.

En lo que respecta a las bases de datos de MICCAI, éstas fueron entrenadas desde cero en la red a fin de ver la importancia que tiene el número de datos de los que se dispone a la hora de evitar el sobrentrenamiento. Observándose así que a mayor número de imágenes que vea la red, mejores resultados se obtendrán en cuanto a métrica se refiere. Estos conjuntos de datos han sido probados en dos *set* de *test* diferentes, uno proporcionado por los propios creadores y con características ligeramente diferentes a las imágenes de *train* y otro sacado mediante una subdivisión de las imágenes de *train*. El motivo de

esta subdivisión no fue otro que el poder aplicar una métrica como el IoU que permita la comparación con otras arquitecturas destinadas a la segmentación semántica.

En cuanto al entrenamiento de la base de datos francesa, éste se realizó mediante *fine tuning* desde el conjunto de datos de intervenciones porcinas de MICCAI. El motivo de esta decisión no fue otro que el de partir de unos pesos obtenidos a partir de un conjunto con mayor número de imágenes y con unas características en cuanto a detección similares a la base de datos de hígados.

Tras el entrenamiento de los diferentes conjuntos sobre la MIS-Net, se procedió a llevar a cabo el mismo proceso pero sobre la UNet y la SegNet a fin de comparar los resultados obtenidos por ellas. En dicha comparación, se ha podido observar como en los tres casos el tiempo de ejecución es superior a los 20 FPS, pudiéndose decir que todas ellas trabajan en tiempo real.

En cuanto a la calidad de los resultados obtenidos, se ha podido comprobar como la UNet se sigue manteniendo como una de las mejores redes en segmentación de imágenes médicas seguida muy de cerca por la MIS-Net, propuesta realizada en este trabajo.

Además, en lo que respecta a las diferentes bases de datos, se ha podido ver la importancia de disponer de un número elevado de datos que permita a la red poder ver una variedad de imágenes durante el entrenamiento a fin de no caer en un sobrentrenamiento de la misma. Aún así, se trata de un problema difícil de afrontar debido al tipo de imágenes que se trata y las diferencias existentes entre ellas, donde aspectos como los brillos de los órganos o los tubos de la operación pueden llevar a confusión entre las posibles clases a segmentar.

Sin embargo, a pesar de no disponer de bases de datos de grandes dimensiones y con una alta variedad de imágenes, se ha conseguido poder diseñar una red que segmente en gran parte diferentes tipos de datos y distinga en ellos las clases más destacables en cuanto a tamaño y dimensiones. Sin duda, dentro de la segmentación semántica aplicada a imágenes médicas queda mucho por investigar si se quiere implementar en un quirófano a fin de ayudar al especialista. Con este trabajo se ha podido ver que con las arquitecturas más famosas dentro de la segmentación se pueden conseguir resultados significativos pero no lo suficiente buenos como para que supongan un aporte en una intervención. Aún así, con una base de datos de mayor dimensiones o técnicas nuevas que ayuden a la generación de escenas sintéticas, el problema podría ser resuelto con mejores resultados. Aunque el empleo de arquitecturas clásicas no asegura unos resultados exitosos, sería conveniente abordar el problema desde nuevos enfoques dentro del campo de las redes neuronales. Permitiendo así que las escenas de un nuevo paciente no incluido dentro del *set* de entrenamiento, pueda obtener unos resultados óptimos y superiores a los que se pueden obtener a día de hoy.

5.2 Líneas futuras

Para finalizar, se dará paso a plantear unas posibles líneas futuras obtenidas tras la realización de este trabajo de fin de grado:

- **Estudio del comportamiento de MIS-Net sobre un conjunto de datos más numeroso.** Debido a la importancia de disponer de un alto número de imágenes, se propone como línea futura el probar MIS-Net en una base de datos de dimensiones como *CityScape* o *ImageNet*.
- **Empleo de redes adversarias generativas.** Debido a la poca existencia de bases de datos de carácter médico de grandes dimensiones, se propone como opción el empleo de redes adversarias generativas. Esta técnica consistiría en el empleo de dos redes neuronales implementadas para la

generación de imágenes sintéticas. Su procedimiento, como bien se detalla en [43] consiste en el empleo de una red para la generación de datos y otra para la evaluación de los mismos. El objetivo principal es enseñar a una de las redes a que sepa tratar los datos sintéticos creado por la otra red como datos originales similares al conjunto de *train*.

Capítulo 6

Presupuesto

Para llevar a cabo la realización de este Trabajo de Fin de Máster han sido necesarios los siguientes recursos citados a continuación, los cuales se podrían desglosar en costes de equipamiento y costes de mano de obra.

6.1 Costes de equipamiento

- **Equipamiento Hardware:**

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal**
Ordenador de sobremesa	1	1000 €	1000 €
Tarjeta gráfica NVIDIA GeForce GTX 1080	1	530 €	530 €
Coste Total HW			1530 €

Tabla 6.1: Coste Hardware

- **Recursos Software:**

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal**
Ubuntu 16.04 LTS	1	0 €	0 €
Librerías utilizadas (Open Source)	varias	0 €	0 €
Licencia Matlab educativa	1	500 €	500 €
Toolbox Matlab	1	incluida dentro de la licencia	-
Software LaTeX	1	0 €	0 €
Coste Total SW			500 €

Tabla 6.2: Coste Software

6.2 Costes de mano de obra

Concepto	Horas	Coste/hora	Subtotal
Etiquetado base de datos	140	8 €	1120 €
Desarrollo software	500	60 €	30000 €
Mecanografiado del documento	200	10 €	2000 €
Coste Total SW			33120 €

Tabla 6.3: Coste de mano de obra

6.3 Costes total

Concepto	Subtotal
Equipamiento Hardware	1530 €
Recursos Software	500 €
Mano de obra	33120 €
Coste Total	35150 €

Tabla 6.4: Coste total

Bibliografía

[1] A. Khan and S. Ravi, "Image segmentation methods: A comparative study," 2013.

[2] N. L. S. Palomino and U. N. R. Concha, "Técnicas de segmentación en procesamiento digital de imágenes," *Revista de investigación de Sistemas e Informática*, vol. 6, no. 2, pp. 9–16, 2009.

[3] "K-means clustering," https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html.

[4] "A practical introduction to deep learning with caffe and python," http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/.

[5] D. J. Matich, "Redes neuronales: Conceptos básicos y aplicaciones," *Universidad Tecnológica Nacional, México*, 2001.

[6] "Convolutional neural networks for visual recognition," http://cs231n.github.io/convolutional-networks/.

[7] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[10] "Segnet: An image-segmentation neural network," https://www.cyberailab.com/home/segnet-an-image-segmentation-neural-network.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[12] "Endoscopic vision challenge," https://endovis.grand-challenge.org.

[13] "Evaluating image segmentation models," https://www.jeremyjordan.me/evaluating-image-segmentation-models/.

[14] "Cirugía laparoscópica," http://www.fecundas.com/cirugiacutea-laparoscoacutepica.html.

[15] O. A. Van der Meijden and M. P. Schijven, "The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: a current review," *Surgical endoscopy*, vol. 23, no. 6, pp. 1180–1190, 2009.

[16] F. Devernay, F. Mourgues, and È. Coste-Manière, "Towards endoscopic augmented reality for robotically assisted minimally invasive cardiac surgery," in *Medical Imaging and Augmented Reality, 2001. Proceedings. International Workshop on.* IEEE, 2001, pp. 16–20.

[17] T. Collins, D. Pizarro, A. Bartoli, M. Canis, and N. Bourdel, "Computer-assisted laparoscopic myomectomy by augmenting the uterus with pre-operative mri data," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Sept 2014, pp. 243–248.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[20] "PÃ¡gina oficial cityscape," https://www.cityscapes-dataset.com.

[21] B. Kayalibay, G. Jensen, and P. van der Smagt, "Cnn-based segmentation of medical imaging data," *arXiv preprint arXiv:1701.03056*, 2017.

[22] S. S. Al-Amri, N. V. Kalyankar *et al.*, "Image segmentation by using threshold techniques," *arXiv preprint arXiv:1005.4020*, 2010.

[23] S. S. Al-Amri, N. Kalyankar, and S. Khamitkar, "Image segmentation by using edge detection," *International journal on computer science and engineering*, vol. 2, no. 3, pp. 804–807, 2010.

[24] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 105–112.

[25] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

[26] ——, "Efficient convnet for real-time semantic segmentation," in *IEEE Intelligent Vehicles Symp.(IV)*, 2017, pp. 1789–1794.

[27] "Resnet, alexnet, vggnet, inception: Understanding various architectures of convolutional networks," https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/.

[28] "Endoscopic vision challenge, instrumental labeling," https://endovissub-instrument.grand-challenge.org/Data/.

[29] "Endoscopic vision challenge, instrumental and organs labeling," https://endovissub2018-roboticscenesegmentation.grand-challenge.org/Data/.

[30] "Image labeler documentation - ofial matlab page," https://es.mathworks.com/help/vision/ref/imagelabeler-app.html?searchHighlight=image%20labeler&s_tid=doc_srchtitle.

[31] "Loss functions: Cross-entropy," https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html.

[32] "Cross-entropy loss funtion," https://stats.stackexchange.com/questions/260505/machine-learning-should-i-use-a-categorical-cross-entropy-or-binary-cross-entro.

[33] "Gaussian blur," https://en.wikipedia.org/wiki/Gaussian_blur.

[34] "Opencv: Smoothing images," https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html.

[35] "Keras documentation: The sequential model api," https://keras.io/models/sequential/.

[36] "Types of optimization algorithms used in neural networks and ways to optimize gradient descent," https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f.

[37] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[39] "Convolutional neural networks for visual recognition: Transferencia de aprendizaje," http://cs231n.github.io/transfer-learning/.

[40] "Estimating an optimal learning rate for a deep neural network," https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0.

[41] "Keras documentation: Callbacks," https://keras.io/callbacks/.

[42] "Intersection over union (iou) for object detection," https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[43] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *CoRR*, vol. abs/1406.2661, 2014. [Online]. Available: http://arxiv.org/abs/1406.2661

Apéndice A

Requisitos mínimos y herramientas

Para garantizar el correcto desarrollo de este trabajo de fin de master, ha sido necesario empleo de los siguientes requisitos mínimos:

- PC compatible
- Tarjeta gráfica NVIDIA 1080
- Sistema operativo Ubuntu 16.04
- Python versión 3.5.2
- Tensorflow
- Librerías de Keras para el tratamiento de Deep Learning
- Librerías OpenCV
- Cuda, versión 8.0.61
- Gestor de versiones Git

A modo de suplemento, se ha hecho uso de otras herramientas necesarias para la elaboración del proyecto como lo son:

- Entorno de desarrollo Pycharm
- Procesado matemático, Matlab 2017-b. Toolbox *ImageLabeler*
- Procesador de textos LaTeX
- Gestor de referencias JabRef